jamk.fi

Firstbeat-datapankin hakutyökalu

Hanna Ahonen

Opinnäytetyö
Toukokuu 2018
Tekniikan ja liikenteen ala
Insinööri (AMK), Ohjelmistotekniikan tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Ahonen, Hanna	Opinnäytetyö, AMK	Toukokuu 2018
	Sivumäärä 47	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Firstbeat-datapankin hakutyökalu

Tutkinto-ohjelma
Ohjelmistotekniikan tutkinto-ohjelma

Työn ohjaaja(t)
Luostarinen Hannu, Hämäläinen Raija

Toimeksiantaja(t)
Firstbeat Technologies Oy

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi Firstbeat Technologies Oy. Firstbeat on hyvinvointiin, kuluttajatuotteisiin ja huippu-urheiluun keskittynyt jyväskyläläinen yritys, jonka hyödyntämä teknologia perustuu sykevälivaihteluun. Yritys tekee vuosittain useita hyvinvointianalyyseja, joista selviävät hyvinvointia tukevat ja haittaavat tekijät elämässä. Näistä kertynyttä mittausdataa kerätään tutkimuskäyttöä varten anonyyminä datapankkiin. Yrityksen työntekijät puolestaan tarvitsevat työssään hyvinvointianalyysistä kertyneitä mittausdatoja, jotka täyttävät tietyt hakukriteerit. Tätä varten Firstbeat tarvitsi hakutyökalun, jolla voi hakea datapankista mittausdataa, joka täyttää halutut hakukriteerit. Hakutyökalun haluttiin olevan selainpohjainen sovellus, joka toimii yrityksen sisäverkossa. Tämän työkalun avulla käyttäjä voi ladata haetun mittauksen tietokoneelleen.

Hakutyökalusta oli jo aikaisemmin tehty vaatimusmäärittely, mutta käyttöliittymäkuvat vaativat päivitystä. Nämä toteutettiin Adobe XD -ohjelman avulla. Käyttöliittymäsuunnittelun jälkeen alkoi varsinainen kehitystyö, (mikä sisälsi myös uuden teknologian opiskelua). Kehitystyö tehtiin Visual Studio Code -lähdekoodieditorilla käyttäen ReactJs JavaScript -kirjastoa. Työn datanhallinnassa käytettiin Redux JavaScript -kirjastoa. Näiden kahden suositun käyttöliittymäteknologian ja muiden teknologioiden lisäksi työssä oli oleellisessa osassa REST API. Työhön tarvittava data saatiin tietokannasta REST:iä hyödyntäen.

Hakutyökalun käyttöliittymä saatiin projektin aikana toteutettua. Toteutuksessa jouduttiin käyttämään mock-dataa, sillä palvelinpuolen toteutus jäi aikataulusta jälkeen. Kokonaisuutena käyttöliittymä on kuitenkin toimiva ja vastaa kaikkia ominaisuuksia. Projektin aikana pystyi myös tekemään vertailua toimeksiantajan käyttämien teknologioiden, GWT:n ja ReactJs:n välillä.

Avainsanat (asiasanat)
ReactJS, Redux, REST, Web-sovelluksen kehitys

Muut tiedot (salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Ahonen, Hanna	Bachelor's thesis	May 2018
		Language of publication: Finnish
	Number of pages 47	Permission for web publication: x

Title of publication

Search tool for Firstbeat databank

Degree programme

Software Engineering

Supervisor(s)

Luostarinen Hannu, Hämäläinen Raija

Assigned by

Firstbeat Technologies Oy

Abstract

This thesis was assigned by Firstbeat Technologies Oy, the main office of which is in Jyväskylä. Firstbeat industry is wellness, sports, and consumer products and the technology used in these products is based heart rate variability. The company produces many lifestyle assessments that produce a complete picture of individual health and performance. The measurement data from these assessments is gathered anonymously to Data Warehouse sever. The Company's employees sometimes need data about these assessments, and the data must fulfil some search conditions. For this purpose, Firstbeat needed a search tool that employees can use to search the measurement data from the Data Warehouse. The search tool must work in the browser environment and it can be used only from the company's intranet. With this tool, a user can download measurements that fulfils the search conditions.

The requirement specification was already carried out; however, the UI specification images needed some updating. These images were created with Adobe XD. After specification, the main development work started including also studying a new technology. The user interface was developed with ReactJs JavaScript library, and Redux JavaScript library was used for the data flow implementations of the work. The UI was created with these two popular web technologies and some other web technologies. The search tool also used REST API to retrieve data from the database.

The user interface of search tool was created during this project. However, the work used some mock data, because REST resources were not finished in time. As a whole, the user interface works correctly and meets all the defined features. During the project, it was also possible to compare between the GWT and the ReactJs technologies. GWT technology is used by the assignor in other web clients.

Keywords/tags (subjects)

ReactJS, Redux, REST, Web application development

Miscellaneous (Confidential information)

Sisältö

Termit ja lyhenteet 5

1 Työn lähtökohdat 9

1.1 Tausta 9

1.2 Firstbeat Technologies Oy 9

1.2.1 Yleistä 9

1.2.2 Hyvinvointianalyysi 10

1.2.3 Bodyguard 2 -mittalaite 11

2 Työn tavoitteet ja suunnittelu 13

2.1 Yleistä 13

2.2 Toiminnalliset vaatimukset 13

2.3 Ei-toiminnalliset vaatimukset 14

2.4 Käyttöliittymäsuunnittelu 15

3 Käyttöliittymän kehitysteknologiat 15

3.1 ReactJs 15

3.1.1 Yleistä 15

3.1.2 Arkkitehtuuri 16

3.1.3 Virtuaalinen DOM 17

3.1.4 Render ja elmänkaarimetodit 17

3.1.5 JSX 18

3.2 Redux 19

3.3 ReactJs ja Redux 21

4 Muut kehitysteknologiat ja työkalut 23

4.1 Teknologiat 23

4.1.1 HTML 23

4.1.2 CSS 23
4.1.3 JavaScript 23
4.1.4 REST 24
4.2 Työkalut 24
4.2.1 Visual Studio Code 24
4.2.2 Redux DevTools Extension 25
4.2.3 Postman 25
4.2.4 Versionhallinta 26

5 Hakutyökalun ominaisuudet 26
5.1 Kirjautumissivu 26
5.2 Hakuehtosivu 27
5.3 Hakutulokset-sivu 29

6 Toteutus 30
6.1 Yleistä 30
6.2 Projektin aloitus 31
6.3 Käyttöliittymän rakenne 32
6.4 REST-rajapintakutsut 33
6.5 Sovelluksen kokonaisrakenne 36

7 Tulokset 37
7.1 Hakutyökaluohjelma 37
7.2 Ohjelman jatkokehitys 38
7.3 React vs. GWT 38

8 Pohdinta 40

Lähteet 42

Liitteet 44

Liite 1. Esimerkki hyvinvointianalyysista saaduista tuloksista 44

Liite 2. Spesifikaatiokuva hakutulokset-taulukosta 46

Liite 3. Sovelluksen hakemistorakenne 47

Kuviot

Kuvio 1. Bodyguard 2 -mittalaite 11

Kuvio 2. Bodyguard 2 -mittalaitteen käyttöohjeet 12

Kuvio 3. NPM-pakettien React, Angular ja Vue latausmäärät kahden viimeisen vuoden aikana 16

Kuvio 4. React-sovelluksen rakenne 17

Kuvio 5. Yleiskatsaus Render-prosessista ja komponentin elämänkaaresta 18

Kuvio 6. JSX-syntaksilla luotu React-elementt 18

Kuvio 7. JSX käännettynä JavaScriptiksi 19

Kuvio 8. Esimerkki storen rakenteesta 19

Kuvio 9. Esimerkki Action Creators -funktiosta 20

Kuvio 10. Esimerkki Rducer-funktiosta 20

Kuvio 11. Redux-tilanhallintaketjun toiminta 21

Kuvio 12. Havainnekuva ReactJs-kirjaston ja Redux-arkkitehturin yhteistoiminnasta 22

Kuvio 13. Redux storen luominen käyttäen hyödyksi Redux DevTools Extension työkalua 25

Kuvio 14. Hakutyökalun kirjautumissivu 27

Kuvio 15. Hakutyökalun hakuehtosivu 28

Kuvio 16. Hakutyökalun hakutulokset-sivu 30

Kuvio 17. packake.json tiedosto ja sen rakenne 32

Kuvio 18. React Routerin määrittely App.js tiedostossa 33

Kuvio 19. Main.js-tiedosto, joka sisältää tiedon navigoitavista komponenteista ja niiden URL-polut 33

Kuvio 20. Action creator-funktio, joka kutsuu REST-rajapintaa 34
Kuvio 21. esimerkki REST-rajapintakutsusta, jonka avulla käyttäjä tunnistetaan 35
Kuvio 22. Helper.js tiedosto virhetilanteiden käsittelyä varten 36
Kuvio 23. Sovelluksen kokonaisrakenne 37
Kuvio 24. Google-hakujen määrää ReactJS ja GWT vuosina 2004-2017 40

Taulukot

Taulukko 1. Hakutyökalun toiminnalliset vaatimukset 14
Taulukko 2. Hakutyöalun ei toiminnalliset vaatimukset 14

Termit ja lyhenteet

API

API on lyhenne ohjelmointirajapinnasta (engl. Application programming interface). Ohjelmointirajapinta tarkoittaa tapaa, jolla eri ohjelmat voivat vaihtaa dataa keskenään eli ns. keskustella.

BMI

BMI eli painoindeksi (engl. Body Mass Index)

Debugata

Debuggaamisella tarkoitetaan virheiden korjaamista. Se myös sisältää virheen etsimisen, sillä virheen aiheuttaja ei ole tiedossa ennen debuggaamista.

DOM

Tapa kuvata esimerkiksi HTML-dokumentin rakenne puurakenteena.

Elektrodi

Tässä työssä elektrodilla tarkoitetaan EKG-elektrodia. Elektrodi on ihoon kiinnitettävä kertakäyttöinen mittausväline, jonka avulla sydämen sykettä pystytään mittamaan.

Elementti

Rakenteisesta dokumentista kuten HTML puhuttaessa elementti tarkoittaa samaa asiaa kuin tagi. Se kertoo, mistä dokumentin rakenne osaa alkaa tai loppuu.

Eväste

Eväste tallentaa pienen määrän dataa käyttäjän tietokoneelle webohjelman toimesta. Seuraavan kerran kun ohjelmaa käytetään, se saa datan käyttäjän koneelta.

Flux

Flux on JavaScript-kirjasto, joka on tarkoitettu sovelluksen tiedonkulun ja arkkitehtuurin hallitsemiseen. Se noudattele MVC-mallia eli malli-näkymä-käsittelijä (engl. model-view-controller) ohjelmistoarkkitehtuuria. Tämän mallin tunnetuin käyttäjä on Facebook.

Funktio

Funktio on aliohjelma eli se tarkoittaa ohjelmoinnissa koodikokonaisuutta, joka suorittaa tietyn toiminnallisuuden.

http-protokolla

http on tiedonsiirrossa käytetty protokolla (engl. Hypertext Transfer Protocol). Asiakasohjelma eli tavallisesti internetselain avaa yhteyden palvelimelle ja lähettää pyynnön datan saamiseksi. Palvelin vastaa joko pyydetyllä datalla tai virheviestillä.

IDE

Ohjelmointiympäristö eli ohjelma, jolla ohjelmoija toteuttaa ohjelmia.

JSON

JSON on tiedostomuoto, joka on tarkoitettu tiedon välitykseen.

Metodi

Ohjelmoinnista puhuttaessa sanaa metodi ja funktio käytetään yleensä ristiin. Kumpikin tarkoittaa aliohjelmaa. (Ks. funktio)

Mock-data

Mock-datalla tarkoitetaan keinotekoista dataa, joka jäljittelee oikeaa dataa. Tällaista dataa käytetään esimerkiksi käyttöliittymien rakentamisessa, jolloin ohjelmiston kehitys ei ole riippuvainen palvelinpuolen toteutuksesta.

Node.js

Node.js on avoimen lähdekoodin sovelluskehys. Sen ansiosta JavaScript-koodia voi suorittaa palvelimella.

NPM-paketti

NPM-paketin avulla voi jakaa avointa lähdekoodia. NPM paketin avulla saa projektiinsa käyttöön valmista toiminnallisuutta, minkä on joku toinen ohjelmistokehittäjä tehnyt.

Objekti

Objekti on kuin taulukko, joka sisältää nimi-arvo-pareja. Esimerkiksi henkilö-objektin kenttiä voisivat olla nimi ja ikä. Kenttien arvot voisivat puolestaan olla Minna ja 50.

Otsake

Otsake-sanalla tarkoitetaan http otsaketta (engl. header). Otsaketiedoissa voi määritellä tarkempia tietoja tehtävästä http-kutsusta.

Parametri

Tarkoittaa esimerkiksi funktiolle käynnistyksen yhteydessä välitettävää tietoa, joka tarvitaan funktion suorittamiseen.

Same-origin policy

Same-origin policy eli saman alkuperän käytäntö on websovelluksissa käytettävä turvallisuuskäytäntö. Palvelimen resursseihin pääsee käsiksi vain samalta palvelimelta. Tämä estää hakkereiden toimintaa.

SDK

SDK (engl. Software Development Kit) on kokoelma ohjelmistokehittämiseen tarvittavia ohjelmapaketteja.

Skripti

Skriptin avulla voi ketjuttaa komentosarjoja, jotka suorittavat toiminnallisuuksia.

Sovelluskehys

Sovelluskehys tarkoittaa ohjelmistotuotetta, joka muodostaa rungon. Tämän päälle varsinainen ohjelma rakennetaan.

Syntaksi

Ohjelmointikielistä puhuttaessa syntaksilla viitataan ohjelmointikielen kielioppisääntöihin.

Tagi

Tagi on tunniste, joka on rakenteisen dokumentin kuten HTML:n osa.

UI/UX suunnittelu

UI tarkoittaa käyttöliittymää (engl. user interface) ja UX tarkoittaa käyttökokemusta (engl. user experience). UI/UX-suunnittelussa suunnitellaan käyttöliittymiä huomioiden käyttäjän tarpeet. Käyttökokemuksesta halutaan mahdollisemman käyttäjäystävällinen.

URL

URL (engl. Uniform Resource Locator) on merkkijono, joka kertoo polun tietylle internetsivulle.

1 Työn lähtökohdat

1.1 Tausta

Työn toimeksiantaja Firstbeat Technologies Oy toteuttaa vuosittain useita hyvinvointianalyysejä asiakkailleen. Hyvinvointianalyysi on kolme vuorokautta kestävä sykeväliä mittaava kokonaisuus. Analyysin avulla voidaan selvittää arjen kuormittavat tekijät ja löytää ratkaisut hyvinvoinnin parantamiseksi. Tällä hetkellä hyvinvointianalyyseista saatu data kerätään anonyyminä talteen tutkimuskäyttöön tarkoitetulle Data Warehouse -palvelimelle. Mittauksen profiilitiedoista poistetaan tiedot, joista henkilö voitaisiin tunnistaa. Vain tutkimus- ja kehitystyön kannalta olennaiset tiedot jäävät profiiliin, kuten ikä, sukupuoli ja paino. (Hyvinvointianalyysi n.d.)

Ohjelmistokehittäjät ja -testaajat sekä muut hyvinvointianalyysin parissa työskentelevät tarvitsevat työssään Hyvinvointianalyysista saatavia mittausdatoja. Usein mittausdatoja halutaan myös seuloa erilaisilla hakukriteereillä, kuten mitattavan henkilön sukupuolella tai iällä. Mittausdatojen seulomiseksi ei ole kuitenkaan rakennettu erillistä hakutyökalua. Hakutyökalun avulla henkilökunnan työ tehostuisi, sillä sopivan mittausdatan etsimiseksi ei tarvitsisi kuluttaa työaikaa. Tällä hetkellä sopivaa mittausdataa joudutaan etsimään tietokoneiden hakemistosta tai mittalaitteista, joilla Hyvinvointianalyysi tehdään.

Tämän perusteella toimeksiantaja tarvitsi hakutyökalun, jonka avulla pystytään seulomaan mittausdataa Data Warehouse -palvelimelta. Hakutyökalun tärkein ominaisuus seulomisen lisäksi on mahdollisuus ladata hakukriteereitä vastaava mittausdata omalle tietokoneelle. Hakutyökalun avulla yrityksen toimintamallit tehostuvat huomattavasti, kun työaikaa ei tarvitse käyttää sopivan mittauksen etsimiseen.

1.2 Firstbeat Technologies Oy

1.2.1 Yleistä

Firstbeat Technologies Oy (myöhemmin Firstbeat) on vuonna 2002 perustettu hyvinvointi- ja urheiluteknologian yritys. Firstbeatin päätoimipiste sijaitsee Jyväskylässä ja

yritys työllistää tänä päivänä noin 100 henkilöä. Firstbeatin toiminta voidaan jakaa kolmeen osa-alueeseen: Kuluttajatuotteet, Hyvinvointianalyysi ja Firstbeat Sports.

Firstbeatin hyödyntämä teknologia pohjautuu Kilpa- ja huippu-urheilun tutkimuskeskuksessa (KIHU) tehtyyn tutkimustyöhön. Jyväskylän yliopistossa käynnistyi 2000-luvulla hanke, jonka tarkoitus oli selvittää, voidaanko huippu-urheilijoille tarkoitettua mittausteknologiaa hyödyntää työntekijöiden stressin ja palautumisen mittaamisessa. Tähän tutkimustyöhön pohjaten Firstbeat on kehittänyt sykevälinvaihteluun perustuvan analyysiin, joka mallintaa kehon toimintoja. (Tarinamme 2018.)

1.2.2 Hyvinvointianalyysi

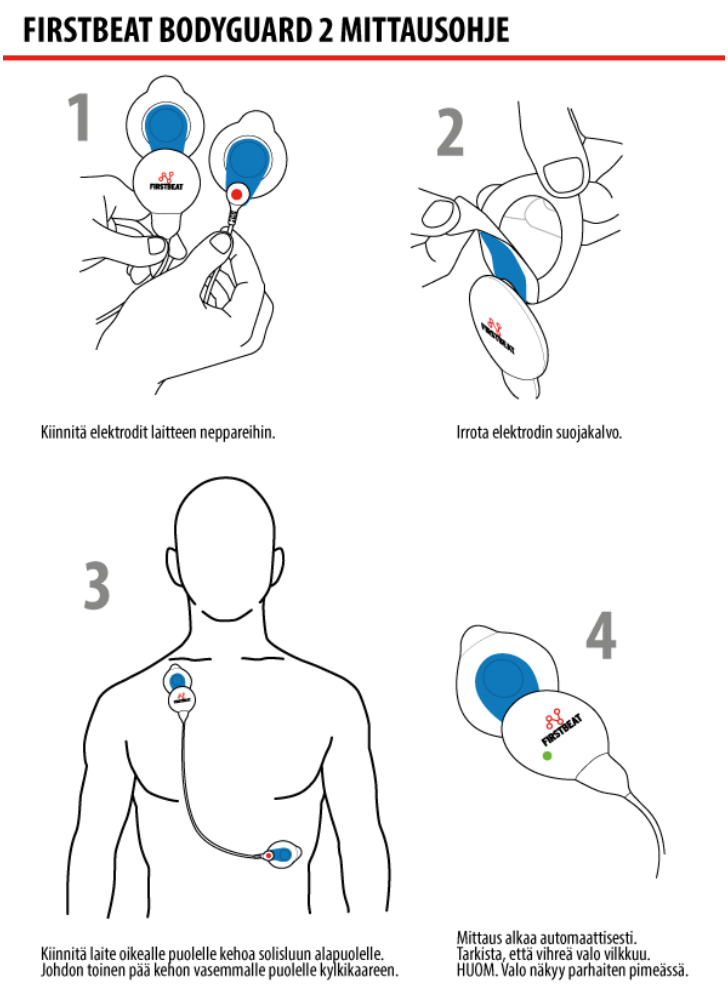
Hyvinvointianalyysin avulla voidaan selvittää, kuinka keho reagoi arjen erilaisissa tilanteissa. Analyysiä varten tarvittava data saadaan kolme vuorokautta kestävästä mittauksesta. Mittausjakso sijoitetaan mitattavan henkilön elämään niin, että se sisältää niin arkipäiviä kuin vapaa-aikaakin. Sykevälianalyysiin ja mittauksen avulla saadaan yksilöllistä tietoa arjen kuormittavista ja palauttavista tekijöistä. Mitattava henkilö saa arvokasta tietoa omasta jaksamisestaan, unen laadustaan ja terveydestään. Analyysi auttaa löytämään kuormittavat tekijät mitattavan henkilön elämässä ja löytämään ratkaisun näihin. (Hyvinvoinnin ammattilaiselle 2018; Työ ja hyvinvointi 2018.)

Ympärivuorokautisen mittausjakson aikana kerättävä data saadaan Firstbeatin Bodyguard 2 -mittalaitteella (ks. kuvio 1). Mittausjakson jälkeen laitteen sisältämä data saadaan purettua USB-portin kautta selainpohjaisessa Hyvinvointianalyysi-ohjelmassa. Data analysoidaan ja analyysin perusteella Hyvinvointianalyysi-ohjelmassa luodaan raportit mittausjakson ajalta. Näistä raporteista mitattava henkilö saa arvokasta tietoa omasta jaksamisestaan ja hyvinvoinnistaan. Esimerkki Hyvinvointianalyysiraportista on liitteenä 1.

Kuvio 1. Bodyguard 2 -mittalaite (Materiaalipankki n.d.)

1.2.3 Bodyguard 2 -mittalaite

Bodyguard 2 -mittalaite on huomaamaton ja ergonominen mittalaite sykevälin mittaamiseen. Mittalaitteeseen kiinnitetään kaksi elektrodia, joiden avulla laite saadaan kiinni kehoon. Laite kiinnitetään oikean puoleisen solisluun alapuolelle ja toinen pää laitteesta asetetaan vasempaan kylkikaareen (ks. kuvio 2). Mittaus alkaa automaattisesti kiinnittämisen jälkeen.



Kuvio 2. Bodyguard 2 -mittalaitteen käyttöohjeet (Hyvinvointianalyysi – ohjeet n.d.)

Bodyguard 2 -mittalaitteen tekniset tiedot:

- Ladattava akku, jonka kesto noin 6 vuorokautta
- Tallennuskapasiteetti noin 20 vuorokautta
- Mittaustarkkuus sykemittauksessa 1 ms (1000 Hz)
- Mittaustarkkuus, liikeanturi:
    - Näytteenottotaajuus 12,5 Hz
    - Resoluutio 8 Bit
    - G-skaala 4 G
- Paino 24 g
- Mittalaite roisketiivis sekä muotoilultaan kevyt ja ergonominen
- Mittaustieto puretaan ja akku ladataan USB-portista ilman erillistä kaapelia.

(Tekniset tiedot n.d.)

2 Työn tavoitteet ja suunnittelu

2.1 Yleistä

Työn tavoitteena oli toteuttaa Firstbeatille Data Warehouse -datapankin hakutyökalu, jolla yrityksen toiminta tehostuisi. Hakutyökalun vaatimusmäärittelyn oli tehnyt hyvinvointiteknologian insinööri Riikka Myötyri vuonna 2016 opinnäytetyössään Hyvinvointianalyysin testidatapankin määrittely ja suunnittelu. Työn tavoitteena oli toteuttaa tämän vaatimuksen mukainen hakutyökalu pienillä rajauksilla sekä parannuksilla. Käytännöstä vaatimusmärittely oli jo niin hyvin tehty, että vain käyttöliittymäkuvat jäivät uudelleen suunniteltavaksi.

Työn osatavoitteena oli myös tutustua ohjelmaa kehittäessä ReactJs-käyttöliittymäkehitykseen periaatteisiin tarkemmin, sillä kyseinen teknologia oli varsin uusi niin toimeksiantajalle kuin työn toteuttajallekin. Tutustumistyön perusteella kyseisen teknologian etuja/haittoja voitiin verrata toimeksiantajalla aikaisemmin käytössä olleeseen Google Web Toolkit -käyttöliittymän (myöhemmin GWT) kehitysteknologiaan.

2.2 Toiminnalliset vaatimukset

Toiminnalliset vaatimukset (ks. taulukko 1) kuvaavat kehitettävän ohjelman ominaisuuksia. Vaatimukset kuvaavat muun muassa, kuinka ohjelman kuulu reagoida käyttäjän syöttämiin syötteisiin tai kuinka ohjelman kuulu toimia erilaisissa tilanteissa. Joissain tapauksissa toiminnalliset vaatimukset kertovat myös, mitä ohjelman ei kuulu tehdä. (Sommerville 2016.)

Taulukko 1. Hakutyökalun toiminnalliset vaatimukset

Vaatimuksen kuvaus
Käyttäjä voi kirjautua järjestelmään
Käyttäjä voi kirjautua järjestelmästä ulos
Käyttäjä voi hakea mittauksen painoindeksin perusteella (BMI)
Käyttäjä voi hakea mittauksen mitattavan henkilön iän perusteella
Käyttäjä voi hakea mittauksen mitattavan henkilön aktiivisuusluokan perusteella
Käyttäjä voi hakea mittauksen mitattavan henkilön sukupuolen perusteella
Käyttäjä voi hakea mittauksen sen keston perusteella
Käyttäjä voi hakea mittaukset, jotka sisältävät liikedataa
Käyttäjä voi hakea mittaukset, jotka sisältävät unijaksoja
Käyttäjä voi ladata haetun mittauksen omalle tietokoneelleen

2.3 Ei-toiminnalliset vaatimukset

Nimensä mukaisesti ei-toiminnalliset vaatimukset (ks. taulukko 2) eivät ole kytköksissä siihen, miten ohjelma toimi. Nämä kuvaavat järjestelmän tarjoamien palveluiden tai toimintojen rajoituksia. Vaatimukset sisältävät mm. tietoturvaan, tehokkuuteen sekä ohjelman kehitykseen liittyviä vaatimuksia. (Sommerville 2016.)

Taulukko 2. Hakutyöalun ei toiminnalliset vaatimukset

Vaatimuksen kuvaus
Sovellusta voi käyttää useampi käyttäjä yhtä aikaa
Sovellus on selainpohjainen
Sovellus toimii yrityksen sisäverkossa
Sovellus on käytettävyydeltään hyvä

2.4 Käyttöliittymäsuunnittelu

Tulevasta käyttöliittymästä oli Riikka Myötyri tehnyt karkeat luonnokset opinnäytetyössään. Näitä hyödyntäen luotiin tarkemmat spesifikaatiokuvat käyttäen hyödyksi Adobe XD -ohjelmaa.

Adobe XD eli Adobe Experience Design CC on uusi konsepti Adoben Creative Cloud -ohjelmistosarjassa. Ohjelma on tarkoitettu PC -ympäristölle ja sillä voi suunnitella käyttöliittymäprototyyppejä. Se on tarkoitettu pääsääntöisesti internetsivujen ja mobiilisovellusten UI/UX-suunnittelun. Ohjelma soveltuu myös muille käyttöalustoille tarkoitettujen ohjelmien suunnitteluun. Ohjelman beta-versio on julkaistu maaliskuussa 2016. (Estrella n.d).

Käyttöliittymän vaatimuksena oli yksinkertaisuus ja helppokäyttöisyys. Myös toimeksiantajalle yleisistä ponnahdusikkunoista pyrittiin pääsemään tässä toteutuksessa eroon, koska ne eivät ole ReactJs kirjastolla toteutetulle käyttöliittymälle tyypillisiä. Ponnahdusikkunoita käytetään tässä ohjelmassa vain virhetilanteissa ja latausikkunassa. Värimaailma tuli yrityksen muiden Hyvinvointianalyysin liittyvien ohjelmien teemasta. Teeman päävärinä on punainen, jota käytetään painikkeissa ja otsikoissa. Muita käytettäviä värejä ovat harmaa, valkoinen ja musta. Esimerkki spesifikaatiokuva on liitteenä 2.

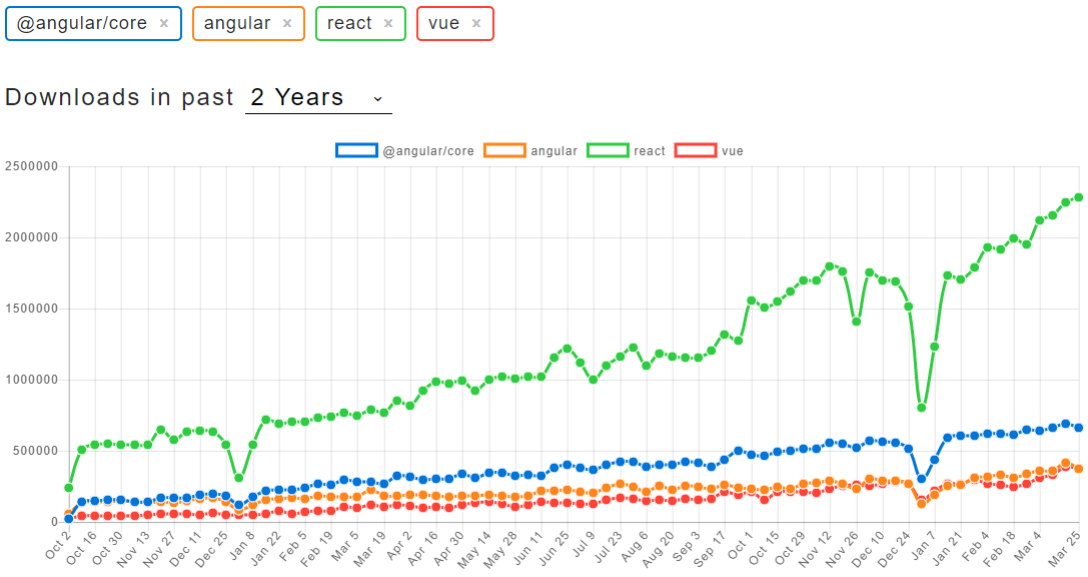
3 Käyttöliittymän kehitysteknologiat

3.1 ReactJs

3.1.1 Yleistä

ReactJs (myöhemmin React) on Facebookin kehittämä avoimen lähdekoodin JavaScript-kirjasto käyttöliittymien kehittämiseen. Sen ensimmäinen avoin versio julkaistiin maaliskuussa 2013. Facebook on kuitenkin käyttänyt sitä jo vuonna 2011 Facebook-seinään, jossa julkaisut esitetään. Reactin pääideana toimivat komponentit. Ohjelmistokehittäjä voi rakentaa uudelleenkäytettäviä käyttöliittymäkomponentteja web- ja mobiilisovelluksiin. Tällä hetkellä React on myös yksi suosituimmista käyttöliittymä JavaScript-kirjastoista. (Papp 2018; ReactJs – Home 2018.)

Kuviosta 4 voi nähdä React NPM -paketin latausmäärän viimeisen kahden vuoden aikana. Kuvio on luotu käyttäen hyödyksi NPM trends -internetsivustoa.

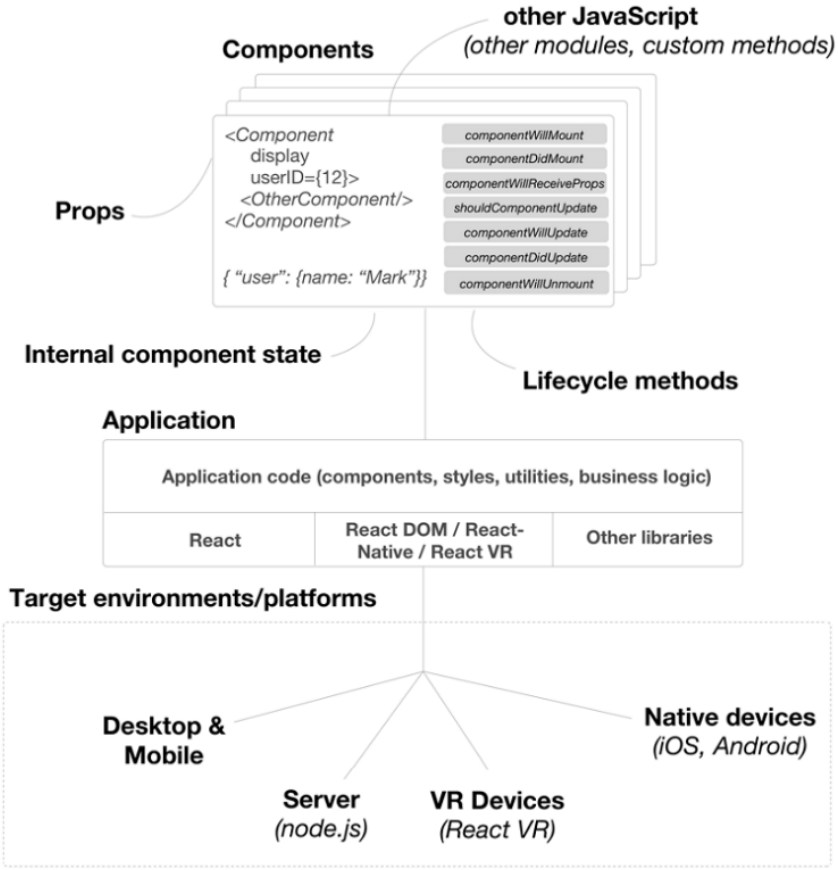


Kuvio 3. NPM-pakettien React, Angular ja Vue latausmäärät kahden viimeisen vuoden aikana

3.1.2 Arkkitehtuuri

React-käyttöliittymäkehityksessä keskeisimmässä osassa ovat komponentit, joita ohjelmistokehittäjä luo. Komponentti voi esimerkiksi olla internetsivun navigointipalkki tai lomake, johon käyttäjä syöttää tietoja. Sovelluksen näkymä koostuu siis näistä komponenteista. Komponentit ovat käytännössä funktioita, joilla on oma sisäinen tilansa (engl. state). Komponentin tilaa voidaan muuttaa syötteiden eli propsien avulla. React tarjoaa komponenttien hallinnalle myös joukon elinkaarimetodeja, joiden avulla komponentin tilanhallinta helpottuu. (Thomas 2017.)

React-sovelluksessa keskeisimmässä osassa ovat React-kirjastot. Web-sovellusta kehittäessä käytetään react-dom-kirjastoa, joka huolehtii sisällön päivittämisestä selaimeen. Muita mahdollisia React-kirjastoja on esimerkiksi React Native, jota käytetään mobiililaitteille suunnatuissa sovelluksissa. React-sovelluksia kehittäessä hyödynnetään myös kolmannen osapuolen kirjastoja. React ei esimerkiksi määrittele, miten sovelluskehittäjän kuuluisi hoitaa http-pyynnöt tai datan käsittely. Tämän voi ohjelmistokehittäjä päättää itse hyödyntämällä ulkopuolisia kirjastoja tai toteuttamalla oman ratkaisun. (Thomas 2017.)



Kuvio 4. React-sovelluksen rakenne (Thomas 2017)

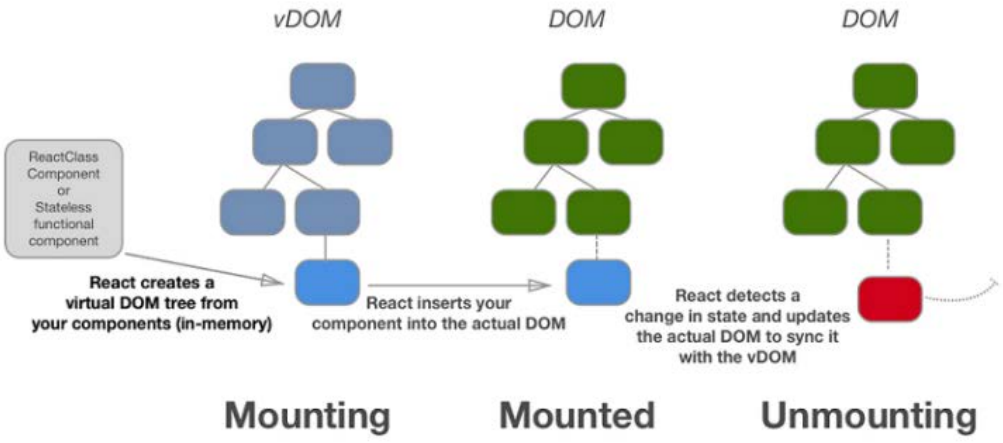
3.1.3 Virtuaalinen DOM

Uudelleen käytettävien komponenttien lisäksi Reactin pääominaisuuksia on virtuaalinen DOM. Virtuaalinen DOM on kuin oikea DOM, mutta se pidetään tietokoneen muistissa. React ylläpitää muistissa olevaa virtuaalista DOM:ia ja päivittää tämän avulla muutokset oikeaan DOM:iin. React pystyy vertailemaan virtuaalisen ja oikean DOM:in eroja, jonka avulla vain muuttuneet kohdat päivittyvät oikeaan DOM:iin. Tämä lisää sovelluksen tehokkuutta huomattavasti, jolloin myös käytettävyys paranee. (Thomas 2017.)

3.1.4 Render ja elämänkaarimetodit

Render-metodi on Reactin päämetodi. Tämän avulla komponentit renderöidään näytölle. Metodi pitää virtuaalisen DOM:in ja oikean DOM:in synkronointia yllä päivittämällä muutokset oikeaan DOM:iin. (Rendering elements n.d.)

Elämänkaarimetodit (engl. lifecycle methods) ovat erityisiä metodeita, jotka suoritetaan React-komponentin tietyssä elinvaiheessa. Nämä metodit voidaan jakaa kahteen pääluokkaan: ’will’-metodit ja ’did’-metodit. Kuten nimi jo kertoo, ’will’-metodit suoritetaan ennen kuin jotain tapahtuu ja ’did’-metodit suoritetaan tapahtuman jälkeen. Nämä kaksi luokkaa voidaan jakaa vielä kolmeen aliluokkaan, jotka kuvaavat, missä vaiheessa komponentin elämää metodi suoritetaan. Mounting-luokan metodit suoritetaan ennen kuin komponentti asetetaan DOM:iin. Update-luokan metodit suoritetaan, kun komponentti vaati päivitystä eli se uudelleen renderöidään. Unmounting-luokan metodit suoritetaan, kun komponentti hävitetään DOM:ista. (Thomas 2017.)



Kuvio 5. Yleiskatsaus Render-prosessista ja komponentin elämänkaaresta (Thomas 2017)

3.1.5 JSX

React-elementtejä käytetään komponenteissa. Näitä on mahdollista luoda kahdella tapaa, joko Reactin createElement-metodin avulla tai käyttäen JSX-syntaksia. JSX on käytännössä JavaScriptin syntaksilisäosa, joka mahdollistaa HTML:n tyylisen koodin kirjoittamisen. Käännösvaiheessa JSX-koodi käännetään kuitenkin tavalliseksi JavaScriptiksi. (Ks. kuviot 6. ja 7.) (JSX In Depth n.d; Thomas 2017.)

```
<MyButton color="blue" shadowSize={2}>
  Click Me
</MyButton>
```

Kuvio 6. JSX-syntaksilla luotu React-elementti (JSX In Depth n.d.)

```
React.createElement(
  MyButton,
  {color: 'blue', shadowSize: 2},
  'Click Me'
)
```

Kuvio 7. JSX käännettynä JavaScriptiksi (JSX In Depth n.d.)

3.2 Redux

Redux on avoimen lähdekoodin JavaScript-kirjasto sovelluksen tilan hallitsemista varten. Se ei kuitenkaan ole perinteinen käyttöliittymäpuolen kirjasto kuten AngularJs, vaan se on enemmän tiedonhallinta-arkkitehtuuri. Yleisimmin Reduxia hyödynnetään ReactJs- tai AngularJs-kirjastojen kanssa. Reduxin hyötyjä on sen pieni koko (vain 2 KB) ja helppokäyttöisyys. (Bachuk 2016.)

Redux pohjautuu Flux-arkkitehtuurimalliin. Toisin kuin Flux-mallissa koko sovelluksen tila (state) pidetään yhdessä paikkaa eli storessa. Storen tila on muuttumaton puurakenteinen objekti (ks. kuvio 8). Jos storen tilaa halutaan muuttaa, siitä luodaan uusi objekti. Tilan muuttamiseen tarvitaan action-objekti sekä reducer-funktio. (Bachuk 2016.)

Kuvio 8. Esimerkki storen rakenteesta

Storen tilaa voidaan muuttaa action-objektien avulla, jotka sisältävät tarvittavan tiedon tilan muokkaamista varten. Actionit voivat sisältää myös hieman logiikkaa, ja niitä hyödynnetään usein monesta paikkaa sovellusta. Tämän vuoksi actionin luonti sisällytetään yleensä funktioon, jota kutsutaan Action Creators -funktioiksi. (Gelman & Dinkevich 2017.)

```
export function setLoginStatus(status) {
    return {
        type: types.SET_LOGIN_STATUS,
        status: status
    }
}
```

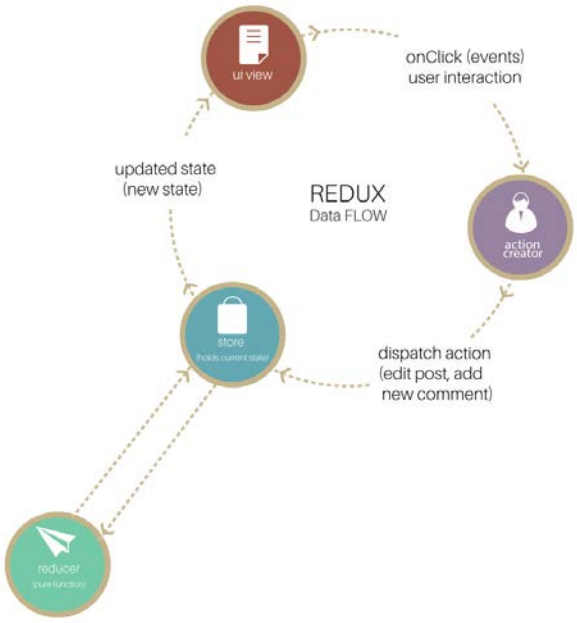
Kuvio 9. Esimerkki Action Creators -funktiosta

Kun actioni lähetetään storelle, se ei sisällä tietoa, miten storen kuuluisi päivittää tila. Tilan päivittämistä varten store kutsuu reducer-funktiota. Redux-sovelluksissa on yleensä yksi reducer-funktio (root reducer), jota store kutsuu sovelluksen nykyisellä tilalla ja saadulla actionilla. Reducer-funktio ei kuitenkaan varsinaisesti muokkaa sovelluksen tilaa, vaan se luo kopion nykyisestä tilasta tarvittavilla päivityksillä. (Gelman & Dinkevich 2017.)

```
export default (state = defaultState, action) => {
    let modalDialog = {};

    switch (action.type) {
        case types.SET_MODAL_MESSAGE:
            modalDialog = {
                message: action.message, leftLabel: null, rightLabel: null,
                leftFunction: null, rightFunction: null
            };
            return { ...state, modalDialog };
        case types.SHOW_MODAL_DIALOG:
            modalDialog = {
                message: action.msg, leftLabel: action.leftLabel, rightLabel: action.rightLabel,
                leftFunction: action.leftFunction, rightFunction: action.rightFunction
            };
            return { ...state, modalDialog };
        case types.LOADING:
            return { ...state, loading: true };
        case types.LOADED:
            return { ...state, loading: false };
        case types.SET_LOGIN_STATUS:
            return { ...state, loginStatus: action.status};
        case types.SET_HOSTNAME:
            return { ...state, hostname: action.hostname };
        default:
            return state;
    }
}
```

Kuvio 10. Esimerkki Reducer-funktiosta

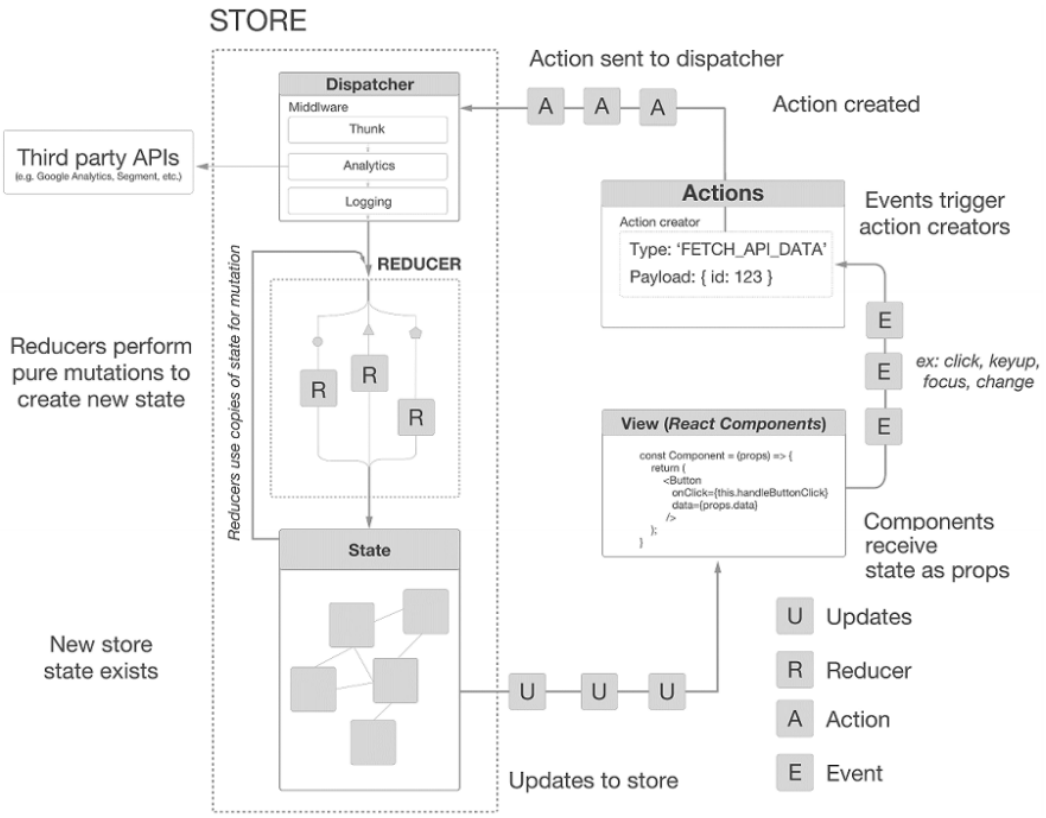
Storen, reducerin ja actioneiden lisäksi Reduxin tilanhallinta-arkkitehtuurissa keskeinen käsite on middleware. Middleware toimii välikätenä actioneiden välittämisessä, mutta on silti Reduxin yksi tärkeimmistä ominaisuuksista. Middlewaren ansiosta on mahdollista suorittaa koodia, ennen kuin store välittää vastaanotetun actionin reducer-funktiolle. Tämä mahdollistaa koodin ketjuttamisen. (Gelman & Dinkevich 2017).



Kuvio 11. Redux-tilanhallintaketjun toiminta (Bachuk 2016)

3.3 ReactJs ja Redux

Kuten edellä mainittiin React-käyttöliittymäkehityksessä voi hyödyntää kolmannen osapuolen kirjastoja. Redux on yksi suosituin tilanhallinta-arkkitehtuuri käytettäväksi Reactin kanssa. Kuviosta 12 nähdään, mitä sovelluksessa tapahtuu, kun käyttäjä painaa käyttöliittymän JSX-syntaksilla kirjoitettua React-elementtiä. Painikkeen painaminen laittaa liikkeelle tapahtumaketjun, joka lopulta päivittää näkymän.



Kuvio 12. Havainnekuva ReactJs-kirjaston ja Redux-arkkitehtuurin yhteistoiminnasta (Thomas 2017)

Painikkeen painaminen ajaa ensin JavaScript-funktion handleButtonClick, joka puolestaan kutsuu action creator -funktiota. Action creator luo JavaScript-objektin, joka välitetään storelle. Store kutsuu reducer-funktiota saadulla objektilla ja storen nykyisellä tilalla. Reducer luo uuden kopion storen tilasta tarvittavilla päivityksillä. React-komponentti kuuntelee storen tilan päivittymistä esimerkiksi elämänkaarimetodin componentWillReceiveProps avulla. Kun komponentti saa syötteenä uuden propsin, sen arvo päivitetään käyttöliittymään Reactin render-funktion ja virtuaalisen DOM:in avulla.

4 Muut kehitysteknologiat ja työkalut

4.1 Teknologiat

4.1.1 HTML

HTML eli Hyper Text Markup Language on eniten käytetty kieli internetsivujen luomiseen. Se kuvaa internetsivun rakennetta elementtien avulla. HTML muistuttaakin hieman syntaksiltaan XML-merkintäkieltä. HTML-kielen on kehittänyt World Wide Webin (WWW) luoja Tim Berners-Lee vuona 1991. Viralliseksi standardiksi HTML muodostui vasta vuona 1995, kun HTML 2.0 julkaistiin. Nykypäivänä käytössä on versio HTML 5. (HTML Tutorial n.d.)

4.1.2 CSS

CSS eli Cascading Style Sheets on tyylittelyyn käytetty kieli. Se kuvaa, kuinka HTML-elementit esitetään näytöllä. Esimerkiksi sillä voi määritellä käytetyn fontin, elementin koon tai värin. CSS:n ensimmäinen versio julkaistiin vuona 1996. Nykyisin on käytössä CSS3-versio, ja sen ominaisuuksia pystyy hyödyntämään moderneissa selaimissa. (CSS 2018.)

4.1.3 JavaScript

JavaScript on tänä päivänä yksi suurimmista ohjelmointikielistä. Sen nousuun on vaikuttanut internetin räjähdysmäinen kasvaminen. JavaScript on vuona 1995 Netscape Communications Corporationin kehittämä webympäristössä käytettävä ohjelmointikieli. Nykypäivänä JavaScriptiä käytetään myös muuallakin kuin webympäristöissä. (Peyrott 2017.)

JavaScript on heikosti tyypitetty kieli eli muuttujien tyyppi ei tarvitse olla tiedossa. Syntaksiltaan JavaScript muistuttaa hieman C -, C++ - ja Java-ohjelmointikieltä. JavaScriptiä ei pidä kuitenkaan sekoittaa Java-ohjelmointikieleen, sillä ne ovat aivan eri kielet, vaikka nimestä voisi ymmärtää toisin. JavaScriptin virallinen nimi onkin ECMAScript, mutta kaupallinen nimi JavaScript on vakiintunut, ja sitä käytetään niin puhekielessä kuin tieteellisissä artikkeleissa. (Flanagan 2006; Peyrott 2017.)

4.1.4 REST

Tietokannan ja websovellusten tarvitsee usein vaihtaa dataa keskenään. Tämä toteutetaan usein Representational State Transfer -teknologian avulla eli tutummin REST -rajapinnan kautta. Tämän työn aikana REST-rajapintojen toteutus ei kuulunut työn toteuttajalle, vaan niiden kehityksen hoiti palvelinpään kehittäjä.

REST on http-protokollaan perustuva ohjelmointirajapintojen toteutusmalli. Sen pääideana ovat resurssit, joilla on URL. Kaikki vuorovaikutukset resurssin kanssa toteutetaan http:n POST-, GET-, PUT- ja DELETE-metodien avulla. REST-rajapintaa kutsuttaessa resurssi voi palauttaa dataa lähes kaikissa formaateissa. Yleisesti kuitenkin käytetään JSON- ja XML-formaatteja, mutta erimerkiksi binaaridata on myös mahdollista. Jos resurssin kutsussa tapahtuu ongelma REST palauttaa http-protokollan mukaisia statuskoodeja kuten 404 tai 500. (Rouse 2017; Sommerville 2016.)

Esimerkiksi ohjelmassa voi olla näkymä asiakkaiden nimistä. Sivun latauksen yhteydessä kutsutaan seuraavanlaista esimerkkiresurssia:

```
www.domain.com/restfulapi/customers.
```

Vastauksena web palvelu palauttaa JSON-muotoista dataa:

```
{ ”customers” : [ ”Name1”,  ”Name2”,  ”Name3” ] }.
```

4.2 Työkalut

4.2.1 Visual Studio Code

Visual Studio Code on Microsoftin kehittämä tehokas mutta kevyt lähdekoodin editoimiseen tarkoitettu sovellus. Se on saatavilla niin Windows-, Linux- kuin macOS-käyttöjärjestelmälle. Asentamisen mukana tulee valmiina tuki JavaScriptille, TypeScriptille ja Node.js:lle. Visual Studio Code ei varsinaisesti ole IDE, vaan se on enemmin tekstieditori. Sovellus kuitenkin sisältää IDE:lle tyypillisiä ominaisuuksia kuten GIT-versionhallinnan tuen ja syntaksin esitäytön. Sovellukseen on myös saatavilla monia lisäosia, joten jokainen pystyy muokkaamaan sovelluksesta käyttötarkoituksiinsa sopivan. Erityisesti se on noussut web-kehittäjien suosioon julkaisunsa jälkeen. (Overview n.d.)

4.2.2 Redux DevTools Extension

Redux DevTools Extension on selaimeen asennettava lisäosa, jolla voi seurata Redux storessa tapahtuvia muutoksia sovelluksen ajon yhteydessä. Työkalu tekee sovelluksen debuggaamisesta helpompaa. Sen avulla pystyy esimerkiksi seuraamaan kutsuttavia actionin luojia ja pysäyttämään sovelluksen ajon tiettyyn kohtaan. Työkalu on saatavilla Google Chromelle ja Mozilla Firefoxille. Selainlisäosan asentamisen lisäksi Reduxin storen luonnin yhteydessä on käytetty NPM-pakettia redux-devtools-extension. Tämän paketin avulla selainsovellus saa tarvittavat tiedot storesta, jotta muutosten seuraaminen onnistuu. Koodissa on myös määritelty, ettei Redux DevTools Extensionia käytetä tuotantoversiossa. (Ks. kuvio 13.)

```
import { createStore, applyMiddleware, compose } from 'redux'
import { composeWithDevTools } from 'redux-devtools-extension';
import thunk from 'redux-thunk';
import reducer from './reducer'

function getComposeEnhancers() {
  if (process.env.NODE_ENV === 'development') {
    return composeWithDevTools;
  } else {
    return compose;
  }
}

const store = createStore(reducer, getComposeEnhancers()(
  applyMiddleware(thunk)
));

export default store;
```

Kuvio 13. Redux storen luominen käyttäen hyödyksi Redux DevTools Extension työkalua

4.2.3 Postman

Postman on Google Chromeen asennettava lisäosa, jonka avulla rajapintojen kanssa työskentely on helpompaa. Postman-sovellus on lähinnä tarkoitettu REST-rajapinnan kanssa työskentelyyn. Mutta sen avulla voi myös esimerkiksi pystyttää mock-rajapintoja. REST-rajapintojen kanssa työskentely on huomattavasti helpompaa Postmanin avulla, sillä rajapintoja voi testata ennen varsinaista toteutusta. Sovelluksen avulla voi tehdä REST-kutsuja tarvittavilla otsaketiedoilla ja muilla parametreillä. Kutsun jälkeen sovellus näyttää, mitä REST-rajapinta palauttaa.

4.2.4 Versionhallinta

Versionhallinta on merkittävässä osassa nykypäivän ohjelmistokehitystä, sillä projektia ei kannata aloittaa ilman varmuuskopiointia. Datan menetys on suuri, jos lähdekoodeja pidetään vain yhdellä tietokoneella. Versionhallinta on myös paljon muutakin kuin tiedostojen varmuuskopiointia. Sen avulla voidaan seurata tiedostoissa tapahtuvia muutoksia ja palata aina edelliseen versioon, jos uusi aiheuttaa ongelmia. (Loeliger & McCullough 2012.)

Tässä projektissa versionhallintatyökaluna käytettiin GIT-järjestelmää. GIT on vuonna 2005 syntynyt versionhallintajärjestelmä. Sen kehitykseen on vaikuttanut Linuxin perustaja Linus Torvalds. GIT syntyi Linuxin kernelin kehitystä varten, ja sen pohjana on käytetty BitKeeper-lähdekoodin hallintajärjestelmää. GIT-järjestelmän syntymän jälkeen siitä on tullut ohjelmistokehityksen johtava versionhallintatyökalu. (Chacon & Straub n.d.)

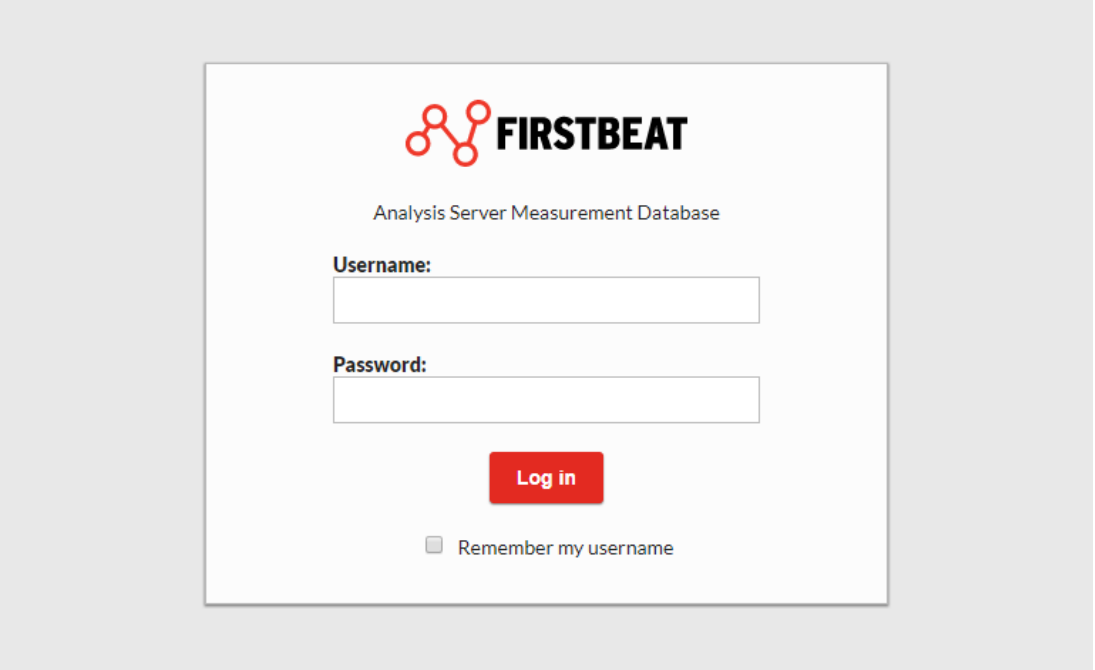
Tämän projektin aikana käytettiin niin graafista käyttöliittymää kuin komentoriviä GITin kanssa työskentelyyn. Graafisena käyttöliittymänä toimi TortoiseGit, joka on Windows-käyttöjärjestelmälle tarkoitettu rajapinta GIT:n hallitsemiseen. TortoiseGit käyttää hyödyksi Windowsin tiedostonhallintaohjelmaa. (About TortoiseGit n.d.)

5 Hakutyökalun ominaisuudet

5.1 Kirjautumissivu

Datapankki hakutyökalun kirjautumissivu (ks. kuvio 14) on yksinkertainen sivu, joka sisältää kentän käyttäjänimen ja salasanan kirjoittamiselle. Sivulla on myös kirjautumispainike ja ”Remember my username” -valintaruutu. Jos kyseinen valintaruutu on valittu päälle järjestelmään kirjauduttaessa, tallennetaan valinta sekä käyttäjänimi selaimen evästeisiin.

Kirjautumispainikkeen painaminen validoi käyttäjän REST resurssin avulla. Jos validointikutsu menee lävitse, käyttäjä ohjataan hakuehtosivulle. Jos validointi ei mene läpi, järjestelmä ilmoittaa joko viallisesta salasanasta tai käyttäjätunnuksesta. Ennen validointia tekstikenttien sisältö tarkistetaan, etteivät kentät ole tyhjät.



Kuvio 14. Hakutyökalun kirjautumissivu

5.2 Hakuehtosivu

Hakuehtosivulla (ks. kuvio 15) käyttäjä voi valita halutut rajaukset mittauksen hakua varten. Rajausvaihtoehtoja ovat sukupuoli, ikä, aktiivisuusluokka, painoindeksi, mittauksen kesto, sisältääkö mittaus harjoittelujakson ja sisältääkö mittaus unijakson. Hakuehdot ovat pudotusvalikoita, joiden sisältö on määritelty vaatimusmäärittelyssä. Pudotusvalikkojen sisältö on seuraavanlainen:

Sukupuoli-pudotusvalikko

- Nainen
- Mies

Ikä-pudotusvalikko

- 18 – 29
- 30 – 39
- 40 – 49
- 50 – 59
- 60+

Aktiivisuusluokka-pudotusvalikko

- 1 – 2
- 3 – 5
- 6 – 8
- 9 – 10

Painoindeksi-pudotusvalikko

- ≤18,4
- 18,5 – 24,9
- 25 – 29,9
- 30 – 34,9
- 35 – 39,9

Mittauksen kesto -pudotusvalikko (h)

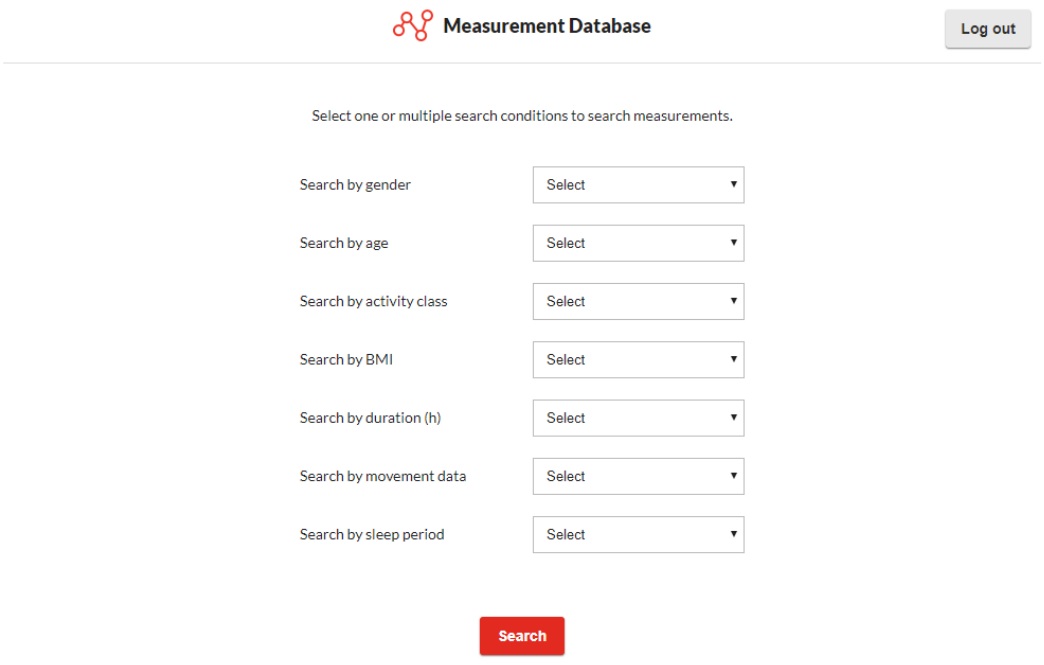
- ≤1
- 1 – 12
- 12 – 36
- ≥72

Liikedata-pudotusvalikko

- Kyllä
- Ei

Unijakso-pudotusvalikko

- Kyllä
- Ei



Kuvio 15. Hakutyökalun hakuehtosivu

Hakuehtosivu sisältää myös hakunapin, joka kutsuu REST-rajapinnan hakuresurssia. Käyttäjän on valittava vähintään yksi hakuehto ennen resurssin kutsumista. Ilman hakuehdon valintaa sovellus avaa virhedialogin ”Selecet at least one search condition.”.

Kun käyttäjä on valinnut hakuehdon/ehdot ja painanut hakupainiketta, ohjelma ohjautuu hakutulokset-sivulle.

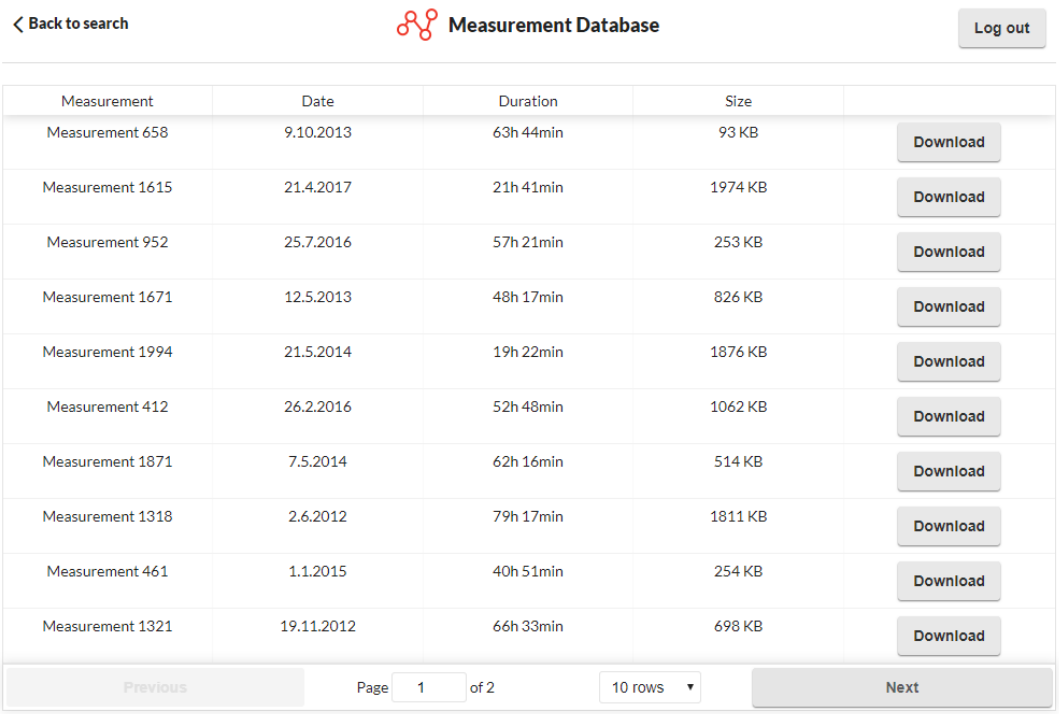
Hakuehtosivu sisältää myös sovelluksen nimen sekä uloskirjautumispainikkeen. Uloskirjautumispainikkeen painaminen hävittää validointitiedot Redux-storesta ja ohjaa näkymän takaisin kirjautumissivulle.

5.3 Hakutulokset-sivu

Hakutulokset-sivu (ks. kuvio 16) sisältää valituilla hakuehdoilla löydetyt tulokset. Tulokset ovat taulukossa, jonka kenttiä ovat Measurement, Date, Duration ja Size. Measurement-kenttä sisältää mittauksen ID:n. Date-kenttä sisältää tiedon, milloin mittaus on aloitettu. Duration-kenttä sisältää mittauksen keston tunteina. Size-kenttä puolestaan sisältää tiedon mittauksen koosta yksikössä KB. Käyttäjä voi myös seuloa hakutuloksia painamalla kentän otsikkoa. Esimerkiksi jos käyttäjä painaa duration-kenttää, hakutulokset järjestetään keston mukaisesti suuruusjärjestykseen. Jokaisen mittauksen vieressä on Download-painike, jota painamalla valittu mittaus ladataan käyttäjän tietokoneelle. Mittausdata ladataan fbe-tiedostomuodossa.

Taulukko näyttää oletuksena 10 kappaletta hakutuloksia, mutta käyttäjä voi muuttaa näkymää taulukon alareunassa olevalla pudotusvalikolla. Käyttäjä voi myös hypätä suoraan tietylle hakutulossivulle syöttämällä halutun sivun numeron taulukon alareunassa olevaan tekstikenttään. Taulukon alareuna sisältää myös ”Next” ja ”Previous” painikkeet, joilla käyttäjä voi vaihtaa hakutulossivua.

Hakutulokset-sivu sisältää myös uloskirjautumispainikkeen, joka ohjaa käyttäjän takaisin kirjautumissivulle. Hakutulokset-sivulla on myös ”Back to search” -painike, joka ohjaa käyttäjän takaisin nollatulle hakuehtosivulle.



Kuvio 16. Hakutyökalun hakutulokset-sivu

6 Toteutus

6.1 Yleistä

Työn toteutusvaiheeseen kuului varsinaisen ohjelmointityön lisäksi testaustyötä ja suunnittelua. Tavallisesti suunnitteluvaiheessa aikataulutetaan ja jaetaan työvastuut kehittäjien kesken. Tämän projektin kulku oli kuitenkin hieman erilainen, kun toteuttajana oli vain yksi henkilö. Suunnitteluvaiheen konkreettisena tuloksena tässä tapauksessa kerättiin tarvittavat tiedot toteutusteknologioista ja mietittiin sovellukselle järkevää rakennetta.

Suunnitteluvaiheen jälkeen alkoi varsinainen toteutus, joka sisälsi varsinaisen ohjelmoinnin lisäksi testausta ja muuta projektin etenemiseen liittyvää työtä. Toteutuksen konkreettisena tuloksena syntyi käyttöliittymä ja sen toiminnallisuus. Ohjelman testaamista teki työn toteuttaja sitä mukaan, kuin sovellus valmistui. Tästä johtuen sovelluksen ensimmäisessä versiossa on todennäköisesti ohjelmointivirheitä, joita toteuttaja ei ole huomannut.

6.2 Projektin aloitus

React-kirjasto vaatii toimiakseen Node.js-palvelinsovelluskehyksen ja NPM-paketinhallintatyökalun asentamista. Tähän löytyy hyviä ohjeita internetistä. Kun nämä on asennettu, voi luoda react-projektin Node.js-komentoriviohjelman avulla. Helpottaakseni react-projektin aloitusta luotiin sovelluksen kansiorakenne ja tarvittavat riippuvuudet NPM:n create-react-app komennon avulla.

```
npm create-react-app projektin-nimi
```

Tämä komento luo Hello World -ohjelman tapaisen sivun, josta on helppo lähteä liikkeelle. Komento luo samalla seuraavanlaisen hakemiston:

```
projektin-nimi
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   └── favicon.ico
│   └── index.html
│   └── manifest.json
└── src
    └── App.css
    └── App.js
    └── App.test.js
    └── index.css
    └── index.js
    └── logo.svg
    └── registerServiceWorker.js
```

React-projektin aloitus vaati todella monen NPM-paketin asentamista. Paketin asentamisen jälkeen kyseisen paketin lähdekoodit ovat sovelluksen juuressa olevan kansion node_modules alla. Näitä lähdekoodeja voi käyttää koodissa import-lauserakenteella. Tieto asennetuista NPM-paketeista tulee myös package.json-tiedostoon, jossa on käytettyjen pakettien nimet ja versionumerot (ks. kuvio 17). Tämän tiedoston avulla voi päivittää uuden version NPM-paketista tai poistaa halutun paketin. Seuraavana esimerkki, kuinka Redux-riippuvuus asennetaan sovellukseen:

```
npm install --save redux

npm install --save react-redux

npm install --save redux-thunk
```

Projektin luomisen ja tarvittavien riippuvuuksien asentamisen jälkeen projektia on mahdollista ajaa seuraavalla komennolla:

```
npm start
```

Komento ajaa käynnistysskriptin, joka on määritelty packake.json-tiedostossa. Skriptin ajon jälkeen projektia voi tarkastella paikallisesti URL-osoitteessa http://localhost:3000 tai skriptiin voi määritellä haluamansa host- ja porttitiedot.

```
{
  "name": "data-warehouse-client",
  "version": "0.1.0",
  "private": true,
  "homepage": "/measurement-database",
  "dependencies": {
    "font-awesome": "^4.7.0",
    "moment": "^2.21.0",
    "namor": "^1.0.1",
    "prop-types": "^15.6.0",
    "react": "^16.2.0",
    "react-dom": "^16.2.0",
    "react-modal": "^3.1.12",
    "react-redux": "^5.0.6",
    "react-router-dom": "^4.2.2",
    "react-table": "^6.7.6",
    "redux": "^3.7.2",
    "redux-thunk": "^2.2.0"
  },
  "devDependencies": {
    "react-scripts": "1.1.0",
    "redux-devtools-extension": "^2.13.2"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  }
}
```

Kuvio 17. packake.json tiedosto ja sen rakenne

6.3 Käyttöliittymän rakenne

Sovelluksen rakennetta miettiessä päädyttiin sovellus jakamaan kolmeen container-komponenttiin, jotka sisältävät pienempiä komponentteja. Container-komponentteina toimivat Login.js, Search.js ja Results.js. Nämä olivat luonnollinen valinta container-komponenteiksi, koska ne ovat sovelluksen kolme päänäkymää. Sovelluksen muita komponentteja on esimerkiksi Header.js, joka sisältää log out -painikkeen ja sovelluksen nimen. Sovelluksen kaikki komponentit ja hakemistorakenteen on esitetty liitteessä 3.

Sovelluksen container-komponenttien eli näkymien välillä liikkuminen on toteutettu react-router-dom NPM-paketin avulla. Routerin avulla voi luoda kokoelman navigoitavista komponenteista. Ulkopuoliselle routerin toiminta vaikuttaa samalta, kuin sovelluksessa olisi useampi HTML-sivu. Todellisuudessa sovellus on niin sanottu single page applikaatio (SPA) eli siinä on vai yksi HTML-sivu. Uutta sisältöä ei missään vaiheessa ladata palvelimelta, vaan käyttäjä pysyy koko ajan samalla sivulla. Vaikka routerin ansiosta sovelluksen URL vaihtuukin näkymien välillä liikkuessa, pysytään koko ajan saman index.html-sivun alaisuudessa. React vain renderöi uuden näkymän index.html tiedostossa olevaan juuri div elementtiin. Kuviot 18. ja 19 selventävät routerin toimintaa.

```
<div className="appContainer">
  {this.props.loginStatus === ''
    ? <Login />
    : (<div>
      <Router history={history}>
        <div className="layoutPanel">
          <Main />
        </div>
      </Router>
    </div>
    )}
</div>
```

Kuvio 18. React Routerin määrittely App.js tiedostossa

```
class Main extends Component {

  render() {
    return (
      <Switch>
        <Route path={locations.search} component={Search} />
        <Route path={locations.results} component={Results} />
        <Route path={locations.notFound} component={NotFound} />
        <Redirect to={locations.notFound}/>
      </Switch>
    );
  }
}
```

Kuvio 19. Main.js-tiedosto, joka sisältää tiedon navigoitavista komponenteista ja niiden URL-polut

6.4 REST-rajapintakutsut

REST-resurssit vaativat aina URL:n, jota kutsua. Tässä tapauksessa, kun websovellus toimii vain yrityksen sisäverkossa, kutsuttava URL täytyy määrittää hosts-tiedostoon.

Tässä tiedostossa voidaan määrittää domain-nimet ja niiden IP-osoitteet. Kutsuttaessa warehouse.firstbeat.fi domain-nimeä, ohjaa hosts-tiedostossa oleva tieto oikeaan sisäverkon IP-osoitteeseen. Windows-käyttöjärjestelmä ympäristössä kyseinen tiedosto löytyy hakemistopolusta %SystemRoot%\System32\drivers\etc\hosts.

REST-ketju lähtee liikkeelle, kun käyttöliittymä kutsuu actionin creator-funktiota. Funktioon on sisällytetty try-catch-rakenne, jotta REST-kutsun aikana tapahtuneet virheet näytettäisiin ponnahdusikkunassa. (Ks. kuvio 20.)

```
export function login(username, password) {
    return function (dispatch) {
        dispatch(loading());
        return isAccountValid(username, password).then((accountStatus) => {
            if (accountStatus === 'Valid') {
                dispatch(setCredentials(username, password));
                dispatch(setLoginStatus('Success'));
                history.push(locations.search);
            } else{
                throw new Error("Invalid credentials");
            }
            dispatch(loaded());
        }).catch((error) => {
            dispatch(loaded());
            dispatch(setModalMessage(error.message));
        });
    };
}
```

Kuvio 20. Action creator -funktio, joka kutsuu REST-rajapintaa

Kuvion 20. action creator -funktio kutsuu REST-rajapintaa, joka on tarkoitettu käyttäjän validoimiseen. Funktion rakenne on hyvin yksinkertainen. Jos REST-rajapinta palauttaa käyttäjänimellä ja salasanalla kutsuttaessa tiedon ’Valid’, asetetaan storeen loginStatuksen tilaksi ’Success’. Tällöin käyttäjä pääsee siirtymään sovelluksen hakusivulle. Muussa tapauksessa näytetään ponnahdusikkunassa virheviesti.

Action creator -funktiot eivät varsinaisesti tee REST-rajapintakutsua, vaan ne kutsuvat erillistä funktiota, joka sisältää varsinaisen kutsun ja virhetilanteiden käsittelyn. Nämä funktiot on toteutettu JavaScriptin Fetch API:n avulla, joka muistuttaa toiminnaltaan monelle web-kehittäjälle tuttua XMLHttpRequestia. Fetch API on kuitenkin huomattavasti tehokkaampi ja tarjoaa uusia ominaisuuksia XMLHttpRequestiin verrattuna. Fetch API hyödyntää kutsuissa Fetch()-metodia ja JavaScriptin promise objektia. (Ks. kuvio 21.)

```
function getHeaders(username, password) {
    return {
        'Accept': 'application/json',
        'Content-Type': 'application/json',
        'Authorization': 'Basic ' + window.btoa(username + ':' + password)
    }
}

function getServiceAddress(){
    return getHostnameFromStore() + serviceAddress;
}

function getHostnameFromStore(){
    return store.getState().hostname;
}

function getHeadersFromStore() {
    return getHeaders(store.getState().user.username, store.getState().user.password);
}


export function isAccountValid(username, password) {
    return new Promise((resolve, reject) => {
        const url = getServiceAddress() + 'me/account-status';
        let headers = getHeaders(username, password);
        headers.Accept = "text/plain";
        fetch(url, {
            headers: headers,
            method: 'GET'
        }).then(function (response) {
            if (response.ok) {
                response.text().then(function (data) {
                    resolve(data); // accountStatus
                });
            } else {
                helper.handleFault(response, reject);
            }
        }).catch(function (error) {
            reject(helper.handleNetworkError(error));
        });
    });
}
```

Kuvio 21. esimerkki REST-rajapintakutsusta, jonka avulla käyttäjä tunnistetaan

Oletuksena Fetch tekee aina GET-pyyntöjä, jos metodille ei määritetä kutsuttavan URL:in lisäksi muuta tieota. Toteutuksessa määritettiin fetch-metodille myös otsake-tiedot, jotka kertovat sallitun vastaus datan muodon ja välittävät autentikointitiedot kutsuttavalle resurssille. Toteutuksessa on myös hyödynnytetty promise-objektin ominaisuuksia. Promisen avulla kutsun toiminnallisuutta on saatu ketjutettua, joka mahdollistaa paremman virhetilanteiden käsittelyn.

REST-kutsun rakenne noudattaa myös try-catch-rakennetta, jotta kutsun aikana tapahtuneet virhetilanteet käsiteltäisiin oikeaoppisesti. Virhetilanteiden käsittelyä var-

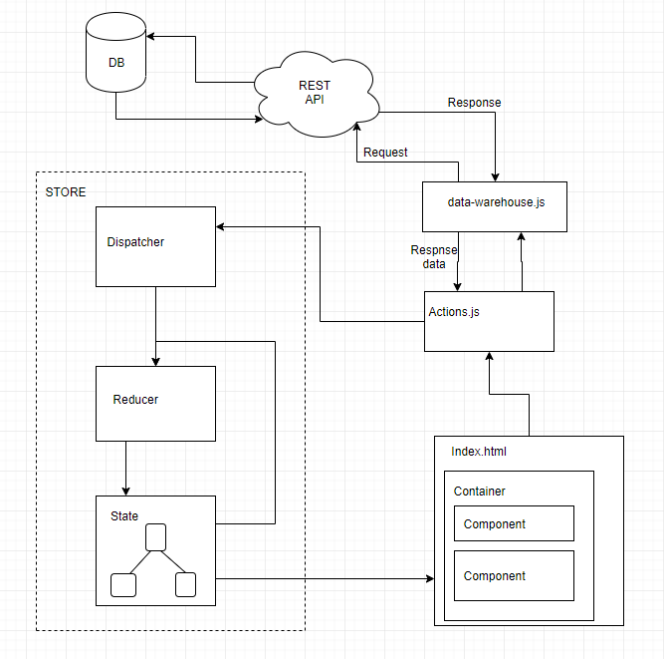
ten sovellukseen on rakennettu helper.js-tiedosto (ks. kuvio 22), jonka avulla virheviesti kirjoitetaan ymmärrettävään muotoon ja näytetään ponnahdusikkunassa. Näin ollen sovellus ei kaadu virhetilanteessa.

```
import moment from 'moment';
import timeUtils from './timeUtils';
import LH from './../localization/localization'

function isServiceFault(status) {
    return status === 400 || status === 401 || status === 404 || status === 409 || status === 500;
}

const helper = {
    handleFault: (response, reject) => {
        if (isServiceFault(response.status)) {
            response.json().then(function (fault) {
                reject(buildServiceError(JSON.stringify(fault)));
            }).catch(function (error) {
                reject(buildNetWorkError(response.status));
            });
        } else {
            reject(buildNetWorkError(response.status));
        }
    },
    handleNetworkError: (status) => {
        return buildNetWorkError(status);
    },
    handleServiceError: (fault) => {
        return buildServiceError(fault);
    },
    handleFaultObject: (fault) => {
        return buildServiceError(JSON.stringify(fault));
    }
}

function buildNetWorkError(error) {
    const timestamp = moment(timeUtils.getTime()).format('YYYY-MM-DD HH:mm:ss');
    return new Error(LH.getText("error.networkError") + '\n\n' + timestamp + (error ? ': ' + error : ''));
}

function buildServiceError(fault) {
    const timestamp = moment(timeUtils.getTime()).format('YYYY-MM-DD HH:mm:ss');
    return new Error(LH.getText("error.serviceError") + '\n\n' + timestamp + ': ' + fault);
}

export default helper;
```

Kuvio 22. Helper.js-tiedosto virhetilanteiden käsittelyä varten

6.5 Sovelluksen kokonaisrakenne

Sovelluksen kokonaisrakennetta tarkastellessa on käyttöliittymän rakenteen lisäksi otettava huomioon, kuinka sovellus viestii REST-rajapinnan kanssa. (Ks. kuvio 23.)



Kuvio 23. Sovelluksen kokonaisrakenne

7 Tulokset

7.1 Hakutyökaluohjelma

Opinnäytetyön konkreettisena tuloksena syntyi käyttöliittymä Hyvinvointianalyysistä saatujen mittausdatojen lataamiseen. Käyttöliittymä vastaa vaatimusmäärittelyä yksinkertaisuudellaan ja helppokäyttöisyydellään. Myös vaaditut hakukriteerit ja kirjautumiseen liittyvät ominaisuudet on toteutettu työhön.

Työn toteutuksessa jouduttiin käyttämään mock-dataa eli ns. keinotekoista dataa, joten tavoitteisiin ei aivan päästy. Projektin aikana palvelinpuolen kehittäjällä oli paljon muitakin työtehtäviä, joten kaikkia REST-resursseja ei ehditty tehdä. Mock-datan avulla käyttöliittymä toimii lähestulkoon suunnitelman mukaan. Mittauksen latauspainike ei tässä tapauksessa aloita oikean mittauksen latausta, vaan se avaa uuden välilehden selaimeen. Tämä toteutettiin näin, koska oikeankin REST-resurssi palauttaisi URL:in, jota kautta lataus alkaisi. Työssä on kuitenkin jo REST-rajapintaan liittyvät käsittelyt, joten työn loppuun viemiseen ei mene kauan, kunhan loputkin REST-resurssit valmistuvat.

Mock-datan käyttö oli toisaalta hyvä, sillä nykyään monia käyttöliittymiä kehitetään mock-datan avulla. Tämän ansiosta käyttöliittymäkehittäjät eivät ole riippuvaisia palvelinpuolen toteutuksesta. Projekti ei jää paikalleen, vaikka palvelinpuolella ei edistyttäisi aikataulussa.

Työn aikana olisi voinut suunnitella sovelluksen arkkitehtuurin paremmin, mikä tulisi helpottamaan sovelluksen jatkokehitystä. Varsinkin jos ohjelman jatkokehityksen tulee toteuttamaan eri henkilö. Jos aloittaisin työn nyt, pilkkoisin komponentteja pienemmiksi. Tämä parantaisi koodin luettavuutta. Tällä hetkellä sovelluksen komponentit ovat turhan isoja, eli ne saattavat sisältää koko näkymän HTML koodin. Tämä on hieman ristiriidassa Reactin uudelleen käytettävän komponenttiperiaatteen kanssa.

7.2 Ohjelman jatkokehitys

Tulevaisuudessa ohjelmaan pitää lisätä admin-tason käyttäjiä ja heille sovellukseen hallintaominaisuuksia käyttäjien lisäämistä ja poistamista varten. Tässä vaiheessa tämä ei ollut vielä tarpeen, sillä sovelluksella on vain muutamia yrityksen sisäisiä käyttäjiä. Tulevaisuudessa jos useampi henkilö alkaa käyttää sovellusta, tarvitaan myös lokitietoja käyttäjien toiminnasta.

Sovellukseen olisi myös hyvä saada lisää hakukriteereitä mittaukselle. Näin ollen entistä tarkempaa datan seulomista voidaan tehdä. Mahdollisia hakukriteereitä voisivat olla askeleiden määrä, energiankulutus ja mitattavan henkilön min/max syke. Mitattavan henkilön profiili- ja mittaustiedoissa on paljon muutakin dataa, jota voi tulevaisuudessa hyödyntää hakukriteerinä.

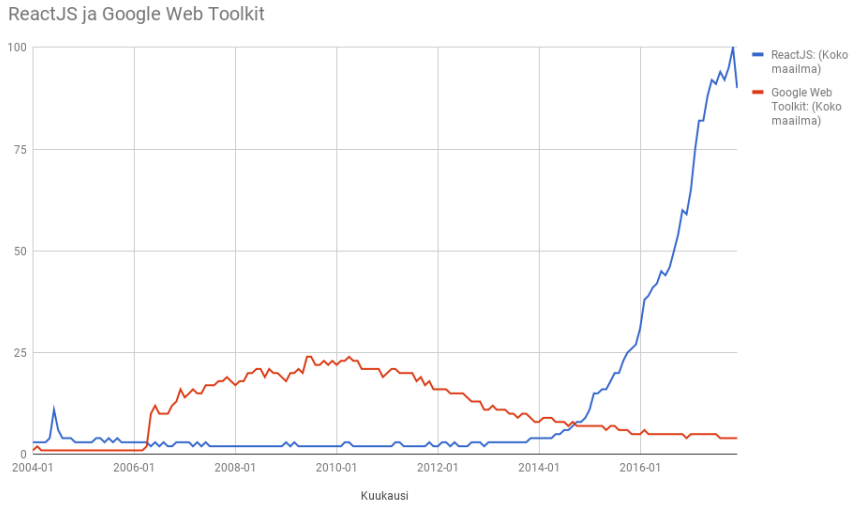
7.3 React vs. GWT

Opinnäytetyön tekemisen ohella olin toimeksiantajalle palkkatöissä. Toimeksiantaja on toteuttanut useita asiakasohjelmia GWT-käyttöliittymän kehitysteknologialla, joten myös GWT-teknologian periaatteet ovat tutut työn toteuttajalle. Tämä mahdollisti React-teknologian etujen/haittojen vertailun GWT-teknologiaan.

GWT on hyvä valinta webkäyttöliittymän kehitysteknologiaksi, jos ohjelmistokehittäjällä ei ole JavaScript-osaamista. GWT-käyttöliittymissä ohjelmointikielenä toimii Java, joten siinä voi hyödyntää monelle ohjelmistokehittäjälle tuttuja Javan arkkitehtuurimalleja, kuten MVP-mallia. Myös yksikkötestit voi kirjoittaa Java-kehittäjille tutulla JUnit-sovelluskehityksellä. Vaikka ohjelmointikielenä GWT-teknologiassa toimiikin Java, käännösvaiheessa koodi käännetään JavaScriptiksi, jota selaimet ymmärtävät. Tämän perusteella GWT on hyvä valinta, jos on aikaisempaa Java-osaamista ja kehitteillä on esimerkiksi yrityskäyttöön tarkoitettu työkalu.

GWT ei ole paras teknologia nykypäivänä muodissa olevien kaupallisten, paljon toiminnallisuutta sisältävien internetsivujen toteuttamiseen. Tämän tyylisillä sivuilla on usein paljon animaatioita ja muita visuaalisia efektejä, joiden toteuttamiseen GWT-teknologialla joutuu näkemään huomattavasti enemmän vaivaa. Reactilla puolestaan visuaaliset efektit on todella helppo toteuttaa CSS3:n avulla tai hyödyntäen erilaisia CSS kirjastoja. React on myös siinä mielessä parempi, että se hyödyntää JSX-teknologiaa, joka tuntee kaikki HTML5-kielestä tutut tagit. HTML-pohjan rakentaminen on siis huomattavasti vaivattomampaa kuin GWT:n UIBinderin avulla.

Kankeuden lisäksi GWT-teknologian ylläpidettävyys on tänä päivänä huono. Internetin eri keskustelupalstoilla käydään kiivasta keskustelua, onko GWT ”kuolemassa pois”. Kun Google 2000-luvun lopulla julkaisi GWT-teknologian, tämä helpotti monen yrityksen toimintaa, kun asiakasohjelman koodin pystyi kirjottamaan samalla kielellä kuin palvelinpuolen. Alussa GWT olikin suosiossa, mutta nykyään JavaScriptin monet sovelluskehykset ovat menneet siitä ohitse. GWT:n SDK-pakettiin ei myöskään ole enää tullut kiivasta tahtia päivityksiä ja uusimmassakin on tunnettuja ohjelmointivirheitä eli bugeja. Edellä mainittujen asioiden perusteella GWT:n suosio on laskussa, varsinkin Reactin suosioon verrattuna (ks. kuvio 24). Reactin suosio on räjähtänyt muutaman viime vuoden aikana, joten React-kirjaston kehitys on tällä hetkellä aktiivista. Sen tuen loppumista ei tarvitse pelätä lähitulevaisuudessa, sillä tunnetut yritykset käyttävät kyseistä teknologiaa.



Kuvio 24. Google-hakujen määrää ReactJS ja GWT vuosina 2004-2017 (Google Trends 2018)

Kaiken edellä mainitun perusteella suosittelen toimeksiantajaa tulevaisuudessa kehittämään käyttöliittymät React-kirjaston avulla. JavaScript-kieltä ei voi tänä päivänä vältellä webkehityksessä, joten React on huomattavasti parempi vaihtoehto kuin GWT. Huonona ominaisuutena React-kirjastosta sanoisin sen sekavuuden. Tietyn asian voi tehdä niin monella tapaa, ja koodi ei ole läheskään yhtä helposti luettavaa kuin Javan järjestelmällinen ja ehkä jopa kurinalainen koodi. Tulevaisuudessa toimeksiantajan pitäisikin keskittyä luomaan React-käyttöliittymäkehitykseen yhteiset periaatteet, jotta jokainen käyttöliittymä noudattaisi samaa rakennetta.

8 Pohdinta

Työ oli hyödyllinen ja opettavainen kokemus uraansa aloittavalle insinöörille. Tämän päivän yksi tärkeimmistä käyttöliittymien kehitysteknologioista tuli tutuksi. Tämä luo hyvän pohjan perehtyä lisää käyttöliittymäkehitykseen ja kehittyä alalla. Teknologian opiskelun lisäksi työ kasvatti henkisesti, sillä projektin kulku oli täysin oma-aloitteisuudesta kiinni.

Työn aikana kohtasin erilaisia ongelmia, mutta näistä selvittiin. Mieleenpainuvin ongelma oli REST-rajapintojen kanssa. Olin työskennellyt REST-rajapintojen kanssa aiemminkin, mutta tällöin ei ollut minkäänlaisia ongelmia. Kaikki sujui siis niin kuin

piti. Vasta tätä työtä toteuttaessa jouduin perehtymään REST-rajapintojen toimintaan, koska kaikki ei sujunut suunnitelmien mukaan. Esimerkiksi kohtasin ongelman selaimien same-origin policyn kanssa. Aluksi REST-rajapintaan ei ollut määritelty CORS-otsaketta (Cross-Origin Resource Sharing). Tämä aiheutti ongelman, koska kutsuin resurssia kehitysvaiheessa localhost:3000 domain nimen alta eli yritin päästä resurssiin käsiksi eri palvelimelta. Tämän ei ole sallittua, ellei CORS-otsakkeessa määrittele sallittua domain-nimeä, jolla saa kutsua resurssia eri palvelimelta.

Kokonaisuutena työ oli ihanteellinen opinnäytetyö, sillä se oli erillinen projekti, jonka toteutusvastuu oli kokonaan työn toteuttajalla. Työ pakotti oma-aloitteisuuteen, mikä kasvatti henkisesti ja toi uskoa omaan tekemiseen. Tämän työn jälkeen käyttöliittymän suunnittelun pystyy viemään jo hieman pidemmälle, kun ymmärtää sovelluksen kokonaisrakenteen. Käyttöliittymä ei enää jää pelkkään toimivaan koodiin, vaan pystyy myös rakentamaan toimivan ja yksinkertaisen arkkitehtuurin käyttöliittymälle.

Lähteet

About TortoiseGit. N.d. TortoiseGit työkalun internetsivut. Viitattu 17.3.2018. https://tortoisegit.org/about/

Bachuk, A. 2016. Redux an Introduction. Smashingmagazine internetsivut. Viitattu 3.3.2018. https://www.smashingmagazine.com/2016/06/an-introduction-to-redux/

Chacon, S. & Straub, B. N.d. Pro Git – Everything you need to know about git. Second Edition.

CSS. 2018. MDN web docs internetsivut. Viitattu 17.2.2018. https://developer.mozilla.org/en-US/docs/Web/CSS

Estrella, S. N.d. Adobe XD vs. Sketch - Which UX Tool is Right for You?. Toptal designers internetsivut. Viitattu 17.3.2018. https://www.toptal.com/designers/ux/adobe-xd-vs-sketch-which-tool

Flanagan, D. 2006. JavaScript: The Definitive Guide. Fifth Edition. Sebastopol: O'Reilly Media.

Gelman, I. & Dinkevich, B. 2017. The Complete Redux Book. Leanpub.

Google Trends. N.d. Googlen internetsivu hakusanojen vertailuun. Viitattu 4.2.2018 https://trends.google.com/trends/

HTML Tutorial. N.d. Tutorialspoint internetsivut. Viitattu 18.3.2018 https://www.tutorialspoint.com/html/index.htm

HTML. 2018. MDN web docs internetsivut. Viitattu 17.2.2018. https://developer.mozilla.org/en-US/docs/Glossary/HTML

Hyvinvoinnin ammattilaiselle. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 28.1.2018. https://www.firstbeat.com/fi/tyo-ja-hyvinvointi/hyvinvoinnin-ammattilaiset/

Hyvinvointianaalyysi. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 12.2.2018. https://www.firstbeat.com/fi/tyo-ja-hyvinvointi/hyvinvointianalyysi/

Hyvinvointianalyysi – ohjeet. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 12.2.2018. https://www.firstbeat.com/fi/tuki/hyvinvointianalyysi/ohjeet/

JSX In Depth. N.d. React internetsivut. Viitattu 3.3. 2018 https://reactjs.org/docs/jsx-in-depth.html

Loeliger, J. & McCullough, M. 2012. Version Control with GIT. Second edition. Sebastopol: O'Reilly Media

Materiaalipankki. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 22.1.2018. https://partners.firstbeat.com/fi/muu-materiaali/markkinointimateriaali/

Overview. N.d. Visual studio coden internetsivut. Viitattu 17.2.2018. https://code.visualstudio.com/docs

Papp, A. 2018. The History of React.js on a Timeline. RisingStack internetsivut. Viitattu 7.4.2018. https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/

Peyrott, S. 2017. A Brief History of JavaScript. Auth0 internetsivut. Viitattu 18.3.2018. https://auth0.com/blog/a-brief-history-of-javascript/

ReactJs – Home. N.d. TutorialsPoint internetsivut. Viitattu 4.2.2018. https://www.tutorialspoint.com/reactjs/index.html

Rendering elements. N.d. React internetsivut. Viitattu 3.3. 2018 https://reactjs.org/docs/rendering-elements.html

Rouse, M. 2017. REST (REpresentational State Transfer). TechTarget internetsivut. Viitattu 18.3.2018. http://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer

Sommerville, I. 2016. Software Engineering. Tenth edition. Pearson Education Limited.

Tarinamme. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 22.1.2018. https://www.firstbeat.com/fi/yritys/tarina/

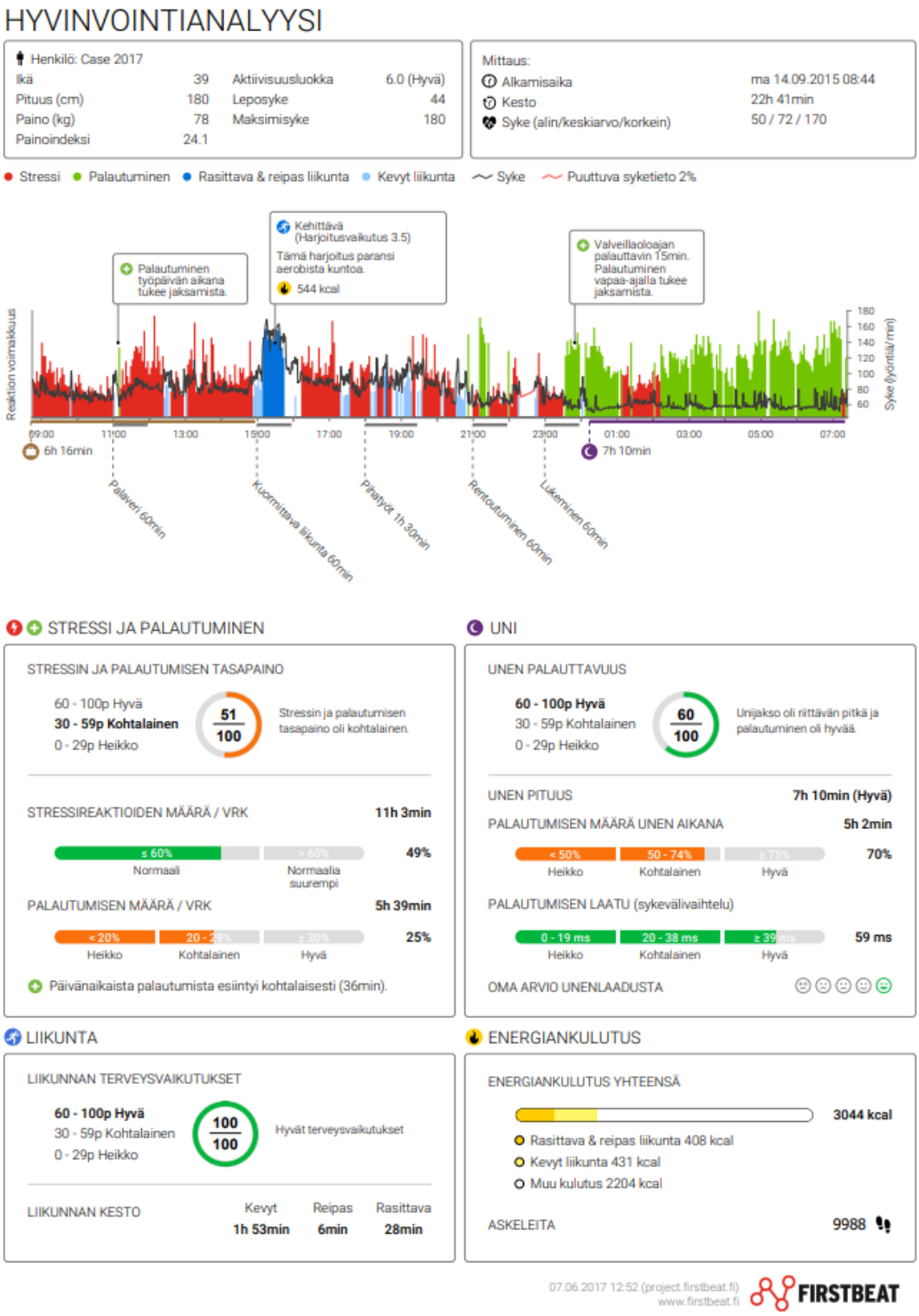
Tekniset tiedot. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 12.2.2018. https://www.firstbeat.com/fi/tyo-ja-hyvinvointi/hyvinvoinnin-ammattilaiset/tekniset-tiedot/

Thomas, M. 2017. MEAP Edition React in Action. Version 9. New York: Manning Publications Co.

Työ ja hyvinvointi. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 28.1.2018. https://www.firstbeat.com/fi/tyo-ja-hyvinvointi/hyvinvoinnin-ammattilaiset/

Liitteet

Liite 1. Esimerkki hyvinvointianalyysista saaduista tuloksista



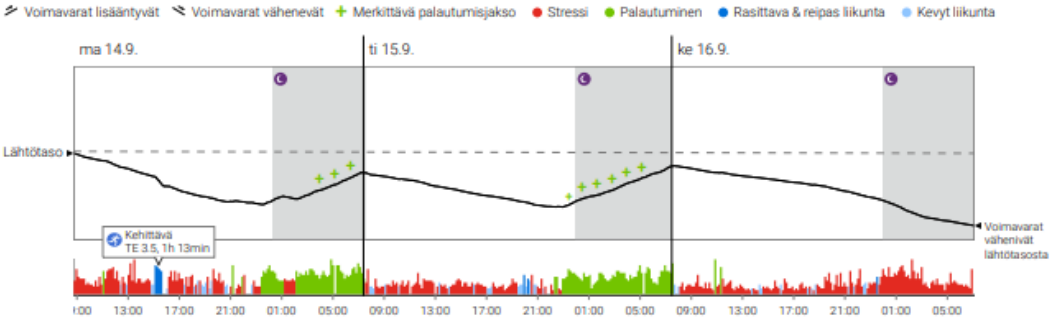
HYVINVOINTIANALYYSIN YHTEENVETO

Henkilö: Case 2017

Ikä	39	Aktiivisuusluokka	6.0 (Hyvä)
Pituus (cm)	180	Leposyke	44
Paino (kg)	78	Maksimisyke	180
Painoindeksi	24.1		

Kartoitus: 14.09.2015 - 16.09.2015
Lisätietoja:
Alkoholia: ke 16.9. (4 annosta)

VOIMAVARAT



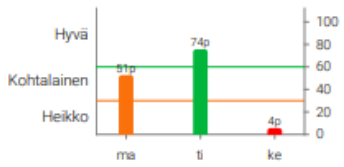
HYVINVOINTIANALYYSIN KOKONAISPISTEET

Tulos perustuu stressin ja palautumisen, unen ja liikunnan mittaustuloksiisi. Näitä osa-alueita parantamalla edistät hyvinvointiasi ja nostat pistemääräsi.

85 - 100p	Erittäin hyvä
60 - 84p	Hyvä
30 - 59p	Kohtalainen
15 - 29p	Heikko
0 - 14p	Erittäin heikko

Hyvinvointianalyysiin osallistuneiden keskiarvo on 55p.

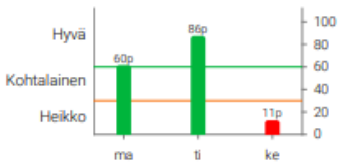
STRESSIN JA PALAUTUMISEN TASAPAINO



Tuloksesi
43
100
KOHTALAINEN

Ikäluokkasi keskiarvo on 55p.

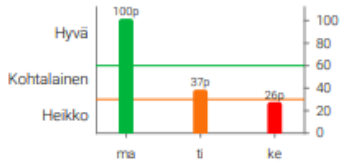
UNEN PALAUTTAVUUS



Tuloksesi
52
100
KOHTALAINEN

Ikäluokkasi keskiarvo on 55p.

LIIKUNNAN TERVEYSVAIKUTUKSET

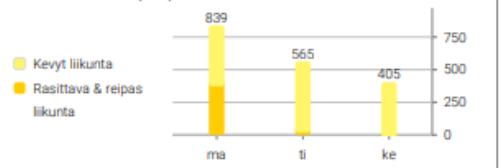


Tuloksesi
61
100
HYVÄ

Mittaukseen sisältyi yksi kuntoa kehittävä harjoitus (harjoitusvaikutus 3.0 tai korkeampi).

Liikunnan tuottamat terveysvaikutukset olivat hyvät yhtenä päivänä.

ENERGIANKULUTUS



	ma	ti	ke
KOKONAISKULUTUS	3043	2856	3241
ASKELEITA	9988	3392	2046

Liite 2. Spesifikaatiokuva hakutulokset-taulukosta

Liite 3. Sovelluksen hakemistorakenne